Probing the Minds of Novice Programmers
Through Guided Learning

A thesis submitted for the degree of
Doctor of Philosophy

Shuhaida Mohamed Shuhidan  B.I.T (Hons.),  M.Sc,
School of Computer Science and Information Technology,
College of Science, Engineering and Health,
RMIT University,
Melbourne, Victoria, Australia.

January, 2012
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Shuhaida Mohamed Shuhidan
School of Computer Science and Information Technology
RMIT University
January, 2012
Acknowledgements

I thank my Lord Allah, to whom I pray and seek guidance, for giving me the strength to complete this memorable expedition.

I offer my greatest gratitude to Margaret Hamilton and Daryl D’Souza, my two beloved supervisors for their encouragement throughout my journey in completing my PhD. Thank you for your informative suggestions and brilliant ideas, for being understanding mentors, giving me the flexibility to work at my own pace and helping me to make it through the ups and downs.

I acknowledge Raymond Lister for his insights as my PhD consultant. A special thanks to Andrew Begel from Microsoft for his guidance in shaping my proposal during the Doctoral Consortium in International Computing Education Research (ICER) 2008. I want to thank Charles Thevathayan and Craig Hamilton, the course instructors for fundamental programming in RMIT University for their help in assisting me to conduct the surveys. Thank you to Nicole Preiss for guiding me with the language structure and grammatical problems. And also not to forget the learning advisors from the Study and Learning centre, RMIT for encouraging me to write and helping me with my English writing.

Thanks to my main sponsor Universiti Teknologi MARA for supporting me all the way in finishing this thesis. I would also want to thank my mentor, Syed Ahmad Aljunid for the words of wisdom and encouragement. A special thanks to Hamdan Abdul Maad for his opinions of the statistical analysis conducted.

I would like to thank my husband, Nirwan Bakri Munir, who is my backbone, for his support and companionship throughout this PhD journey. Adam, my only son, hardly knows what I have been doing spending most of my time in front of the computer but is there to give me the strength to keep on going.

Not to forget my parents, Shuhidan Arifin and Maimun Pandak Dali, for their unconditional love, supports and attention. I dedicate the thesis as a gift for them. Also, many thanks to my siblings Shazalina Shuhidan, Shariza Shuhidan, Shuhaifi Shuhidan, Shariman Shuhidan, especially to Shamila Shuhidan and her family, and also my family-in-law for their ongoing support.

A special note of thanks goes out to all the respondents in the study, without whom this research would not have been completed. Not to forget all my friends who supported me along the journey to complete this thesis.
Credits

Portions of the material in this thesis have previously appeared in the following publications:


This research was undertaken with a scholarship provided by the Malaysian Ministry of Higher Education and Universiti Teknologi MARA (UiTM).
Contents

Abstract

1 Introduction
  1.1 Research Problems .................................................. 5
  1.2 Research Objectives .................................................. 5
    1.2.1 How might one classify multiple choice questions in summative assessment for novice programming? ............ 7
    1.2.2 What are the programming instructors’ views of multiple choice questions in summative assessment? .......... 7
    1.2.3 How can a guided learning tool aid novices in learning programming? ........................................... 7
  1.3 Thesis Structure ...................................................... 7

2 Learning to Program
  2.1 Novice Programmers .................................................. 12
  2.2 Challenges to Learning Programming ................................ 13
    2.2.1 Novice Programmers Learning Complexities .................. 16
    2.2.2 Instructor Teaching Approaches ................................ 18
    2.2.3 Nature of the Programming Course’s Contents ............... 20
  2.3 Learning Programming Aids ......................................... 22
    2.3.1 Learning Through Exercises ................................... 22
    2.3.2 Visualisation may Boost Learning ............................. 23
    2.3.3 Discussion on Learning Tools .................................. 24
  2.4 Study of Assessment ................................................ 24
    2.4.1 Formative Assessment ........................................... 25
    2.4.2 Summative Assessment ........................................... 26
3 Cognitive Theories of Learning

3.1 Cognitive Process ................................................................. 32
3.2 Bloom’s Taxonomy ................................................................. 33
  3.2.1 Original Bloom’s Taxonomy .................................................. 33
  3.2.2 Revised Bloom’s Taxonomy .................................................. 34
  3.2.3 Comparison of Bloom’s Taxonomy [Bloom and Krathwohl, 1956] and
       Revised Bloom’s Taxonomy [Anderson et al., 2001] ...................... 36
  3.2.4 Application of Bloom’s Taxonomy to the Computer Science Area ... 37
3.3 SOLO Taxonomy ................................................................. 39
  3.3.1 Application of SOLO Taxonomy in Computer Science Area .......... 41
3.4 Constructivist Theory ............................................................... 41
3.5 Learning Styles ................................................................. 43
3.6 Theories of Learning for Programming Courses ................................ 44
3.7 Summary .................................................. 45

4 Using Action Research to Understand Learning Difficulties

4.1 Action Research ................................................................. 48
4.2 Applying Action Research Approach ........................................... 50
4.3 Action Research: Cycle 1 ....................................................... 51
4.4 Action Research: Cycle 2 ....................................................... 52
4.5 Action Research: Cycle 3 ....................................................... 53
4.6 Action Research: Cycle 4 ....................................................... 54
4.7 Action Research: Cycle 5 ....................................................... 54
4.8 Respondent Profiles ............................................................... 55
  4.8.1 Novice ................................................................. 55
  4.8.2 Instructor ............................................................... 56
4.9 Details of Programming Courses ................................................ 57
4.10 Descriptions of the Summative Assessment .................................... 57
4.11 Descriptions of Instructor Survey ............................................. 59
4.12 Method of Data Analysis ....................................................... 59
5 Novice Responses to Summative Assessment

5.1 Motivation: Analysis of Multiple Choice Question in Summative Assessment 65
5.2 A Study of Multiple Choice Questions in Summative Assessment 66
  5.2.1 Overall Responses to Multiple Choice Question 67
  5.2.2 Individual Responses to Multiple Choice Questions 68
  5.2.3 Analysing Multiple Choice Questions Based on the Modified Version of Bloom’s Taxonomy 70
  Question 5 71
  Question 11 72
  Question 14 72
  5.2.4 Novice Responses to Multiple Choice Questions 73
  Question 5 74
  Question 11 75
  Question 14 76
  5.2.5 Classification of Multiple Choice Questions Based on the Modified Version of Bloom’s Taxonomy 77

5.3 Reflections 80
5.4 Classification of Summative Assessment Based on Learning Taxonomies 81
  5.4.1 Application of Bloom’s Taxonomy to Multiple Choice Questions 82
  Question 9 83
  Question 15 84
  5.4.2 Instructor Levels of Complexity 85
  5.4.3 Novice Levels of Difficulty 86
  5.4.4 Index of Discrimination 88
  5.4.5 Item Analysis 89
  5.4.6 Data Analysis 94
  5.4.7 Categorisation of Code Writing Question 95
  5.4.8 Application of SOLO Taxonomy to Short Answer Question 97
  Sample of Relational Model Response 98
CONTENTS

Sample of Multistructural Model Response ......................... 99
Sample of Unistructural Model Response ......................... 99
Sample of Prestructural Model Response ....................... 100
Example of No Attempt or Totally Wrong Model Response .... 100
5.5 Discussion ................................................................... 100
5.6 Summary ..................................................................... 102

6 Instructor Perspectives of Multiple Choice Questions ......... 104
6.1 Motivation: Instructor Insights into Multiple Choice Questions in Summative Assessments ........................................... 105
6.2 Instructor Survey ........................................................... 105
  6.2.1 Section A ................................................................. 107
    Question 1 ..................................................................... 107
    Question 2 ..................................................................... 107
    Question 3 ..................................................................... 108
    Question 4 ..................................................................... 108
    Question 5 ..................................................................... 109
    Question 6 ..................................................................... 110
    Question 7 ..................................................................... 110
  6.2.2 Section B ................................................................. 110
    Question 8 ..................................................................... 111
    Question 9 ..................................................................... 112
    Question 10 .................................................................... 113
    Question 11 .................................................................... 114
6.3 Criteria for Instructor Levels of Complexity .................... 115
6.4 Instructor Survey Responses ........................................... 116
  6.4.1 Instructor Background and Experiences ..................... 116
  6.4.2 Instructor Views of Summative Assessment ................ 118
  6.4.3 Instructor Views of Multiple Choice Questions .......... 121
  6.4.4 Instructor Evaluations of Multiple Choice Questions .... 126
6.5 Data Analysis .................................................................. 132
6.6 Discussion .................................................................... 136
6.7 Summary .................................................................... 138
7 Guided Learning Tool for Novice Programming

7.1 Motivation: A Guided Learning Approach to Aid the Learning of Programming

7.2 Guided Learning A

7.2.1 Section A: Debugging Questions

7.2.2 Section B: Multiple Choice Questions

Question 1
Question 2
Question 3
Question 4
Question 5
Question 6
Question 7

7.3 Guided Learning B

7.3.1 Section A

7.3.2 Section B

7.3.3 Levels of Complexity for Multiple Choice Questions

Question 12
Question 20

7.3.4 Section C

7.3.5 Guided Learning B: Getting It All Together

7.4 Summary

8 Evaluation of Guided Learning Tool

8.1 Surveys for the Guided Learning Tools

8.2 Responses to Guided Learning A

8.2.1 Section A: Debugging Question

8.2.2 Section B: Multiple Choice Questions

8.2.3 Novice Suggestions for Guided Learning A

8.3 Reflections

8.3.1 Instruction

8.3.2 Example on Each Topic

8.3.3 Webpage Design

8.4 Cognitive Process Representation Framework

8.5 Responses to Guided Learning B
8.6 Survey Responses .................................................. 179
8.7 Novice Performances in Programming Assessment ............... 182
8.8 Discussion .......................................................... 185
8.9 Summary ............................................................ 187

9 Conclusions ......................................................... 188
9.1 Research Question 1: How might one classify multiple choice questions in summative assessment for novice programming? .................. 189
9.2 Research Question 2: What are the programming instructors’ views of multiple choice questions in summative assessment? ............... 189
9.3 Research Question 3: How can a guided learning tool aid novices in learning programming? ........................................ 191
9.4 Conclusion ......................................................... 194
9.5 Future Works ....................................................... 195
9.6 Concluding Remarks .............................................. 196

A Ethics Approval ....................................................... 197

B Biggs’ Survey ........................................................ 199

C Novice Perspectives Survey ......................................... 205
C.1 Examination Questions for Semester 2, 2007 ....................... 206
  C.1.1 Part 1 ............................................................... 206
C.2 Individual Responses to Multiple Choice Questions ............... 213
C.3 Examination Questions for Semester 1, 2008 ....................... 215
  C.3.1 Part 1 ............................................................... 215
  C.3.2 Part 2 ............................................................... 223

D Instructor Views Questionnaire ..................................... 225
D.1 Instructor Questionnaire ........................................... 226
D.2 Instructor Responses ............................................... 232

E Guided Learning A ................................................... 246

F Guided Learning B .................................................... 252

Bibliography ............................................................ 260
List of Figures

1.1 The Thesis Structure .......................................................... 8
2.1 Programming 1 Students’ Results for 2009 .............................. 14
2.2 Pedagogical Relations in the Didactic Triangle [Kansanen, 1999] .... 16
2.3 The Task Typology [Carbone, 2007] ...................................... 19
3.1 Bloom’s Taxonomy: Cognitive Domain [Bloom and Krathwohl, 1956] 34
3.2 Revised Bloom’s Taxonomy [Anderson et al., 2001] (Cognitive Domain) 36
4.1 Model of Action Research ...................................................... 49
6.1 Outline for Instructor Survey ............................................... 106
6.2 Instructor Responses to Question 8 Based on Syntax Knowledge, Semantic
Knowledge, Problem Solving Skill and Level of Difficulty .................. 126
6.3 Instructor Responses to Question 9 Based on Syntax, Semantic Knowledge,
Problem Solving Skill and Level of Difficulty ................................ 128
6.4 Instructor Responses to Question 10 Based on Syntax, Semantic Knowledge,
Problem Solving Skill and Level of Difficulty .............................. 128
6.5 Instructor Responses to Question 11 Based on Syntax, Semantic Knowledge,
Problem Solving Skill and Level of Difficulty .............................. 129
7.1 Applying Bloom’s Taxonomy to the Guided Learning A .............. 144
7.2 Example of Debugging Question, Section A in Guided Learning A .... 146
7.3 Example of Multiple Choice Question, Section B in Guided Learning A .... 147
7.4 Applying Bloom’s Taxonomy to the Guided Learning B .............. 151
7.5 Week 1 Content of the Guided Learning B: Question 1 .................. 153
7.6 Week 1 Content of the Guided Learning B: Question 2 .................. 154
# LIST OF FIGURES

7.7  The Flow of the Guided Learning B ........................................ 158  
7.8  Screenshot 1 for Guided Learning B, Section B: Question 1 ............ 160  
7.9  Screenshot 2 for Guided Learning B, Section B: Question 2 ............ 161  

8.1  Debugging Question: Correct Responses from Novices .................. 167  
8.2  Multiple Choice Questions: Correct Responses from Novices .......... 168  
8.3  Novices Individual Responses to Question 5 ............................ 170  
8.4  Instructions Added to Guided Learning B .................................. 172  
8.5  Guided Learning B: Topic 2 Week 2 ....................................... 175  
8.6  Novices’ Results for Programming 1 Course .............................. 184  
8.7  Novices’ Results in Java for Programmers Course ....................... 185  

9.1  Cognitive Processes Representation When Learning To Program Using Guided Learning Approach and Tool ............................ 193  
9.2  Relational Level of SOLO Taxonomy ....................................... 193  

F.1  Guided Learning B: Topic 1 of Week 1 ................................... 253  
F.2  Guided Learning B: Topic 2 of Week 2 ................................... 254  
F.3  Guided Learning B: Topic 3 of Week 3 ................................... 255  
F.4  Guided Learning B: Topic 4 of Week 4 ................................... 256  
F.5  Guided Learning B: Other Useful Information ............................ 257  
F.6  Guided Learning B: Instructions .......................................... 258  
F.7  Guided Learning B: Survey .................................................. 259
List of Tables

2.1 Descriptions for Table 2.1 ................................. 15
3.1 Bloom’s Taxonomy and Descriptions (Sourced from [Isaacs, 1996] as referred to [Bloom and Krathwohl, 1956]) ......................... 35
3.2 SOLO Taxonomy .................................................. 39
3.3 SOLO Taxonomy [Biggs et al., 1999] ............................. 40
3.4 SOLO Taxonomy Applied to Learning of Programming [Whalley et al., 2006] 41
3.5 SOLO Categories for Code Writing Task [Lister et al., 2010] ....................... 42
3.6 Responses to Biggs’ Revised Two-Factor Study Process Questionnaire (R-SPQ-2F) .................................................. 44
3.7 Research Groups in Computer Science Education Area ................................. 45
4.1 Description of the Cycles based on Action Research Methodology ............. 50
4.2 The First Cycle ...................................................... 51
4.3 The Second Cycle .................................................... 52
4.4 The Third Cycle ...................................................... 53
4.5 The Fourth Cycle .................................................... 54
4.6 The Fifth Cycle ...................................................... 55
4.7 Programming Courses .............................................. 57
4.8 The Strength and Direction of Correlation Coefficient Classifications by Rowntree [Rowntree, 1981] ........................................ 61
5.1 Overall Responses to Multiple Choice Questions ................................. 67
5.2 Individual Responses to Multiple Choice Questions who Scored 0.88 Proficiency 69
5.3 Individual Responses to Multiple Choice Questions who Scored 0.31 Proficiency 69
5.4 Learning Taxonomies based on Revised Bloom’s Taxonomy [Anderson et al., 2001] and our own Modified Version of Bloom’s Taxonomy 

5.5 Analysis of the Novice Responses for Question 5

5.6 Analysis of the Novice Responses for Question 11

5.7 Analysis of the Novice Responses for Question 14

5.8 Classification of Bloom’s Taxonomy to Multiple Choice Questions

5.9 Multiple Choice Questions in Level 2 (Understand) of Bloom’s Taxonomy

5.10 Multiple Choice Questions in Level 3 (Apply Execute) of Bloom’s Taxonomy

5.11 Multiple Choice Questions in Level 4 (Apply Implement) of Bloom’s Taxonomy

5.12 Correlations Coefficient ($\rho$) Between Correct Responses and Bloom’s Taxonomy

5.13 Correlations Coefficient ($\rho$) Between Correct Responses and Bloom’s Taxonomy

5.14 Bloom’s Taxonomy and the Number of Multiple Choice Questions in Each Level

5.15 Instructor Levels of Complexity

5.16 Novice Levels of Difficulty

5.17 Classifications of Multiple Choice Questions Based on Bloom’s Taxonomy, Instructor Levels of Complexity and Novice Levels of Difficulty

5.18 Index of Discrimination (ID)

5.19 Item Analysis on Question 10

5.20 Item Analysis on Question 11

5.21 Item Analysis on Question 17

5.22 Item Analysis on Question 5

5.23 Item Analysis on Question 16

5.24 Correlations Coefficient ($\rho$) Between Pairwise Bloom’s Taxonomy, Instructor Level of Complexity (Instructor), Novice Level of Difficulty (Novice) and Index of Discrimination

5.25 Category and Descriptions of SOLO Taxonomy (Improvised based on [Biggs, 2008a;b; James, 2011])

5.26 SOLO Taxonomy Applied to Responses for the Short Written Code Segment

6.1 Descriptions of Instructors Grouped by Semesters of Experience in Teaching (Teaching) and in Devising Examination Questions (Devising Question)

6.2 Instructor Views of Summative Assessment

6.3 Other Responses to the Factors Instructors Consider When They Devise Questions (Question 6)
LIST OF TABLES

6.4 Question 3: Why Do You Use Multiple Choice Questions for Summative Assessment? ................................................................. 121
6.5 Reasons Instructors Use Multiple Choice Questions .................................................. 122
6.6 Question 4: How Confident Are You That Multiple Choice Questions In The Final Exam Will... ................................................... 123
6.7 Question 5: How Do You Devise Distractors? .......................................................... 125
6.8 Frequency of Data for Levels of Skills and Knowledge Tested in Questions 8, 9, 10 and 11 ................................................................. 127
6.9 Mean for Levels of Skills and Knowledge Tested in Questions 8, 9, 10 and 11 ........ 131
6.10 Range of the Means for Each Category ................................................................. 131
6.11 Correlations Between Variables Measured for Questions 8 ................................. 133
6.12 Correlations Between Variables Measured for Questions 9 ................................. 133
6.13 Correlations Between Variables Measured for Questions 10 ............................... 133
6.14 Correlations Between Variables Measured for Questions 11 ............................... 134
6.15 Correlation between Experience in Teaching and Experience in Devising Exam Questions with the Four Criteria (Syntax Knowledge (SynK), Semantic Knowledge (SemK), Problem Solving Skills (PSS), Level of Difficulty (LD)) for Questions 8, 9, 10 and 11 ................................................................. 135
7.1 Levels of Complexity for Question 12 and 20 .......................................................... 157
8.1 Respondent Profile ................................................................................................. 165
8.2 Novice Responses to Multiple Choice Questions .................................................. 167
8.3 Novices Correct Responses to Question 5 ............................................................ 168
8.4 Individual Correct Responses to Question 5 ......................................................... 169
8.5 Guide for Figure 8.3 .............................................................................................. 170
8.6 Guided Learning A: Visual Representation of Individual Novice Cognitive Processes Based on Bloom’s Taxonomy ................................. 176
8.7 Guided Learning B: Visual Representation of Individual Novice Cognitive Processes Based on Bloom’s Taxonomy ................................. 178
8.8 Responses to Survey (Part 1) from P1G2 and JPG2 groups .................................. 180
8.9 Responses to Survey (Part 2) from P1G2 and JPG2 groups .................................. 181
8.10 Division of Marks Contributed to Final Grade ...................................................... 183
8.11 Descriptions for Figure 8.6 and Figure 8.7 ........................................................... 183
8.12 Distribution of Novices’ Performances .......................... 184

C.1 Individual Responses to Multiple Choice Questions (Table 1) ............ 213
C.2 Individual Responses to Multiple Choice Questions (Table 2) ............ 214
C.3 Individual Responses to Multiple Choice Questions (Table 3) ............ 215
Abstract

Learning to program is known to be difficult and problematic for a significant number of novice programmers. The problem has generated interest in a range of enquiries and has given impetus to the need for a teaching-research nexus, to provide a better understanding of novice programming problems. Novices are sometimes either unable to comprehend a range of fundamental programming concepts or carry misunderstandings and misconceptions about programming well into their first semester, leading to failures in summative assessment.

Our methodology involved is action research, conducted over five cycles. The first two cycles involve our study of novice responses to summative assessment. In the third cycle we study the instructor perspectives to summative assessment and particularly into multiple choice questions. In the fourth and fifth cycles we detail the development and evaluation of the guided learning tool. The development of the guided learning tool is conducted in two cycles to allow us to reflect and consider the novice responses and suggestions, and provide improvement in the following cycle of action research.

We analyse two sets of summative assessments in introductory programming courses in order to understand the answers provided by novices and the types of errors they typically make. We classify the examination questions using the Bloom’s Taxonomy and the SOLO Taxonomy. In this thesis, we focus our analysis on multiple choice questions. Most of the multiple choice questions tested in summative assessments used for novices are in the lower levels of Bloom’s Taxonomy. We conduct item analysis for each question and calculate the index of discrimination to describe the responses of different groups of novices. We discuss the novice levels of difficulty to analyse novice responses to the multiple choice questions. We also define and discuss measures such as instructor levels of complexity to understand the intentions of the instructors when testing novices via summative assessments.

We conduct further research on instructor perspectives as we find that there are inadequacies in the levels of complexity which they perceive to exist in exam questions. Instructor
perceptions are important because the preparation of all forms of assessment is driven by their perceptions of student learning of the programming concepts based on the course syllabus. This in turn may yield instructor perspectives of summative assessment that do not necessarily correlate with student abilities or expectations or beliefs about what they have been taught. We present the results of our research around instructor perspectives of summative assessment for the novice programmers. Both quantitative and qualitative data have been obtained via survey responses to targeted examination questions from programming instructors with varying teaching experiences.

A great deal of our research centres around the use of multiple choice questions in programming assessments. Multiple choice questions are a popular assessment tool to help novices familiarise and consolidate their understanding of fundamental concepts. Such questions are also typically used in the “traditional practice” or “rote learning” contexts during the formative stages of learning. Our findings encourage us to design and develop a guided learning approach to identify cognitive lapses in learning programming using multiple choice questions as the test instrument.

We propose a guided learning tool within the learning context of novice programmers who are university students with little or no programming background, enrolled in a Computer Science course. We apply Bloom’s Taxonomy to classify the tasks in the guided learning tool. We also introduce a framework to represent cognitive stages for novices when they answer the question. The cognitive stages are classified using Bloom’s Taxonomy. The outcome of this framework can be measured using the SOLO Taxonomy, with the expectation that novices will progress up to the relational level, whereby they can remember, understand, apply and connect all the components learnt by the end of their fundamental programming course. This may ease their progress to the more advanced programming courses.

The guided learning tool is developed around the idea of probing the minds of novices as they use the tool to learn programming. This provides for an adaptive, learner-centric experience, allowing the student a self-directed learning environment. The tool, combined with the framework, allows instructors and novices to monitor novice progress and identify the strong and weak links between the components learned. Instructors may benefit by improving their teaching, as the guided learning tool helps them to identify “what” and “where” the learning difficulties are, and not take for granted the areas or components of learning that they believe the novices have easily understood. Novices may identify their weak components and work on the particular topics to improve their skills in learning to program.
Finally, we report the evaluation of our guided learning approach and tool. Novices are happy to trial and use the guided learning tool. In the survey, we received approximately half of the responses mentioning that the questions in the guided learning tool help them to remember and understand programming concepts, which support our goal of the guided learning tool, that is to aid novices to remember and understand programming concepts. Overall, we hope that the guided learning tool may prevent students from falling into the cognitive traps that often ensnare novice programmers.
Chapter 1

Introduction

Hello World, Hello Programming

The “Hello World” program (as below) is typically the first common program introduced to learners. Based on personal experience, the entire first week can be spent on learning how to print out the two magic words, “Hello World” on the screen.

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The difficult task is to understand the program as a whole, with different new jargon and concepts to become familiar with. There are at least 14 different new words and eight symbols to remember and relate to concepts and their roles, in order to produce a working program. Additionally, the symbols carry their own meanings. If one is missing or is out of place, an error message will be produced. If the error message is not “novice friendly”, this can often lead to frustration and cause de-motivation to the learner. Furthermore, there are programming concepts underlying this program for novices to understand, including: class, object-oriented, access specifier, return type, method, class library, parameters, print statements, to name a few. This problem is explained in a broad scope in the next section.
CHAPTER 1. INTRODUCTION

1.1 Research Problems

We define novices as beginners of learning computer programming. We focus on novices who are university level students enrolled in foundation programming courses. Novice programmers have limited time to attain a full understanding of the program elements and the processes of how a program works. They are expected to be able to write a complete working program plus deal with the new terminology introduced in their first few weeks. This requires novices to have the ability to gain knowledge of basic terminology and to understand the usage of the terminology as a component of a program. All these may cause their thinking processes to overload.

Therefore, research into the challenges of teaching and learning computer programming to novices has long received attention, from those who wish to better understand first year tertiary programming students’ difficulties in conquering introductory programming concepts.

In this thesis, we address novice programming difficulties in the context of a traditional delivery model of teaching and learning. That is, one that uses lectures, tutorials, laboratories to teach programming and assessments to test the students’ abilities.

The course materials are lecture notes, tutorials and laboratory exercises. The lecture materials are delivered to a large number of students each week. This scenario does not support personalised learning and students are not able to learn at their own pace. Assessments are conducted in the form of formative and summative tasks which may consist of assignments, tests and final examination. We will discuss in detail the course delivery in Section 2.4.

Our research focuses on formative assessment because it may help students learn. Sets of questions may be given to students, to increase their level of understanding of the topics. This can happen with or without points or marks being awarded. Formative assessment can be conducted in various formats and on different platforms or environments. The selection of the format for formative, and also summative assessments is important because it contributes to the students’ grades and these reflect the students’ ability to learn programming, directly or indirectly. We highlight our objectives in this research in the next section.

1.2 Research Objectives

Programming is a core subject in the Computer Science area. Basically, programming courses involve learning specific languages to instruct the computer to perform certain tasks. One
has to learn the syntax of the language and have strong problem solving skills to “interact” with the computer.

However, not all novice programmers are able to master the skills to read and write computer programs. Some have difficulties in passing the fundamental programming course. In addition, the course is often a prerequisite for any further study, so if it is failed, it will delay the student’s completion of the course and their degree.

Student performance is often measured through assessments. Therefore, it is important to devise a proper assessment that is able to portray students’ abilities to learn and not obstruct their progression to the next phase of the course. The other factor that relates to student achievements is the instructor. Instructor opinions matter as they are responsible for designing the course and the assessments. They are also partially responsible if the course has a high failure rate.

We aim to develop a platform, in the form of formative assessment to aid novice programmers in learning computer programming. A well organised formative assessment may enable students to explore learning efficiently. We aim to probe the minds of students who are beginners in learning programming through a set of exercises with varying degrees of difficulties.

We noted that there are existing learning tools to aid novice programmers, however we would like to propose a learning tool that focuses on de-constructing the learning complexity using the Bloom’s Taxonomy. We analysed the summative assessment questions and the students’ responses based on the existing learning taxonomies. In addition we propose new classifications for assessing the multiple choice questions. We also seek instructors’ opinions regarding the summative assessment as a whole, but focus on multiple choice questions as a test instrument in programming assessment.

Therefore, as the output, we propose a guided learning approach to aid novices to learn programming. We develop a tool that supports the learning approach that uses multiple choice questions as a formative assessment to encourage novice programmers to practise the jargon (syntax) of the language and to understand basic programming processes. We explore the use of multiple choice questions to aid novices to learn programming. Multiple choice questions encourage prompt and direct answers, therefore, they may be less likely to de-motivate learners to find the answer.

In order to achieve the research objectives, we outline the specific research questions addressed in this thesis as follows.
1.2.1 How might one classify multiple choice questions in summative assessment for novice programming?

Our research interest is to help novice programmers learn better. Our first approach to achieve this aim is to look at the root of the problem. We study novice programmers’ responses through a summative assessment of a basic programming course. We aim to produce a classification of the questions asked in the examinations and to study the responses to the questions. We make use of existing learning taxonomies (which will be further discussed and extended in Chapter 3) to classify the questions.

1.2.2 What are the programming instructors’ views of multiple choice questions in summative assessment?

Instructors play an important role in developing assessments to measure a novice’s ability to learn programming. We survey programming instructors to look at their insights into summative assessments and specifically focus to the multiple choice questions that they use to test their students in summative assessments. We also ask the instructors to evaluate four specific programming questions.

1.2.3 How can a guided learning tool aid novices in learning programming?

Last but not least, we develop a guided learning tool to be tested by novices. The guided learning tool uses multiple choice questions in a formative assessment. It contains a series of programming exercises that cover the first four sections of learning fundamental programming. We use the findings from our research questions 1 and 2 to aid in developing the guided learning tool. Results of the evaluation of the guided learning tool will be presented in quantitative and qualitative form.

1.3 Thesis Structure

In previous sections we outlined our research problems and objectives. In this section, we explain the structure of the thesis in order to meet our objectives. The thesis begins with a brief explanation of the needs that motivate us to conduct this research. As we have highlighted in our research questions, our major contributions aim to improve the learning of programming for novices through the classifications of programming tasks as the input for the development of a guided learning tool.
There are nine chapters in this thesis. The overall thesis structure and the connection between chapters are presented in Figure 1.1.

The brief explanations for the chapters and their contents are as follows;

**Chapter 1: Introduction**

This first chapter highlights the background of the research, emphasising the research problems and subsequently outlining the objectives of the research and the research questions.
CHAPTER 1. INTRODUCTION

Chapter 2: Learning To Program
The second chapter explores the literature in the context of novices in learning programming. These include challenges to learning programming, existing tools that aid novices to learn programming, study of assessments of novice programmers and related research in programming errors.

Chapter 3: Cognitive Theories of Learning
The third chapter presents the descriptions of the theories of education in general and we apply these theories as a means to improve the learning of programming. Thus, we discuss the concepts of mental models of the constructs of programming, cognitive perspectives in learning to program, the Bloom’s Taxonomy, the SOLO Taxonomy, constructivist theory and learning styles that all of which may influence novices in their studying. We also list the group of researchers who applied theories of learning to programming courses.

Chapter 4: Using Action Research to Understand Learning Difficulties
The fourth chapter explains the research approach of this thesis. Action research methodology is employed and is conducted in five cycles. The target respondents are novice programmers and programming instructors. These cycles are planned and explained in Chapter 4. We also describe the respondents, the details of the programming courses, the descriptions of the programming assessment and the instructor survey, and the method of data analysis.

Chapter 5: Novice Responses to Summative Assessment
The fifth chapter presents a study of summative assessment for novice programmers, in particular analysing their responses to multiple choice questions. We find that in a programming assessment, there is a lack of classifications of the difficulty level of a test item. Therefore, we make use of existing learning taxonomies (the Bloom’s Taxonomy and the SOLO Taxonomy) to classify the questions. We propose the instructor levels of complexity and novice levels of difficulty to assist instructors and novices classifying programming questions and the responses in summative assessments. We also calculate the item analysis and index of discrimination to the multiple choice questions.

Chapter 6: Instructor Perspectives Towards Multiple Choice Questions
The sixth chapter presents the results of the programming instructors’ survey regarding multiple choice questions in summative assessment. We distributed the survey across a number
of countries and we detail the responses in qualitative and quantitative analysis.

Chapter 7: Guided Learning Tool for Novice Programming
The seventh chapter highlights our culmination from the major contributions from chapters 2, 3, 4, 5 and 6. We propose the guided learning tool and base our approach on the framework we developed from referring to the revised Bloom’s Taxonomy and the hierarchy of the cognitive processes involved in the programming tasks. We develop the guided learning tool in two cycles of action research.

Chapter 8: Evaluation of the Guided Learning Tool
The eighth chapter evaluates our guided learning approach and tool. We also highlight our framework for the guided learning tool to represent the novices’ cognitive processes.

Chapter 9: Conclusions and Future Works
Finally, in the ninth chapter, we summarise the answers to our research questions and highlight the interesting findings and key conclusions of the research. This chapter also provides some recommendations for future research.
Chapter 2

Learning to Program

Our research aims to contribute to a teaching and learning methodology by establishing a strong foundation of essential, fundamental concepts taught in the early stages of a programming course. This chapter presents related research that is aligned with such interest and also distinguishes our own contributions to this area of research. This chapter aims to survey and to address the context of the research problems presented in Chapter 1.

In order to address the research problems, we first present our clear definition of what we mean by novice programmers (Section 2.1). Next, we classify and discuss the challenges faced by the novices (Section 2.2.1) and instructors (Section 2.2.2) in fundamental programming courses. We also discuss the complexity of the learning contents (Section 2.2.3) that may contribute to novice ability to program.

Instructors may use learning tools to aid their student’s learning. There are many learning tools developed to facilitate learning in programming courses. We list and discuss existing tools that support learning programming (Section 2.3). These learning tools are grouped into two categories: exercise tools (Section 2.3.1) and visualisation tools (Section 2.3.2). We also provide discussions on these learning tools (Section 2.3.3).

Our main goal of the research is to propose a suitable assessment tool that is capable of guiding novice to overcome learning difficulties in programming. Therefore we cover the study of assessments that are often used to measure student ability in programming courses (Section 2.4). Novice successes are measured by their performance in assessments, including formative (Section 2.4.1) and summative (Section 2.4.2) assessment. We place emphasis on the use of multiple choice questions as the test instrument (Section 2.4.3).

We also discuss studies of common errors that novices make in writing a computer pro-
CHAPTER 2. LEARNING TO PROGRAM

gram (Section 2.5). These common errors may be used to design exercises to support novice to overcome the difficulties of learning to program.

2.1 Novice Programmers

Our research focuses on novice programmers (also referred to as novices) who are gaining knowledge of programming. Previous researchers have defined a novice as a person who is in the first stages of becoming a programmer [Bonar and Soloway, 1983; Thomas et al., 2004]. In our context, a novice is a university level, undergraduate or postgraduate student learning fundamental programming, with little or no prior programming knowledge. Novices are often first year computer science or cognate disciplines students who are required to have programming as a significant component of their curriculum. Students transitioning from foundation level or an information technology diploma have often been exposed to introductory programming concepts, without necessarily, being actively engaged in programming and are thus considered novices also. Henceforth, we use the term novices and students interchangeably, with the above definition.

Typically, novices start off with limited surface knowledge and disorganised learning. Moreover, they lack the ability to form accurate and detailed mental models of programming. Many novices fail to apply syntax and semantic knowledge of programming correctly and tend to approach programs “line by line” rather than by using meaningful program “chunks” or structures [Winslow, 1996; Mead et al., 2006]. In related research, Lister stated that novices have better reading skills than writing skills in relation to programming [Lister, 2007; 2008], which indicates there may be hierarchical stages in learning to program. Therefore, this research attempt to classify the questions in summative assessment in order to understand its complexity based on the instructors perspectives. We aim to identify the levels of cognition required to solve the programming questions.

Research has stated that the accepted normal period for a novice to become an expert programmer is ten years [Winslow, 1996]. Expert programmers have higher levels of abstraction based on a pattern of years of experience [Mead et al., 2006]. In this research, instead of focusing on accelerating progression from novices to experts, we are trying to ensure that novices are not hindered by long held misunderstandings of programming concepts. We focus on the concept of practising fundamental programming tasks to minimise novice misunderstanding.

In related research, Robins et al. has described the learning practices of effective and
ineffective novices [Robins et al., 2003]. Effective novices learn to program without excessive effort or assistance. Ineffective novices do not learn, or do so only after inordinate effort and personal attention. Ineffective novices may be deterred by the difficulty of the programming courses. We will discuss these learning difficulties in the next section.

2.2 Challenges to Learning Programming

Programming is one of the difficult courses offered in computer science [Rodrigo et al., 2009]. It is known for being difficult to learn [Robins et al., 2003], which results in poor retention in class [Bornat et al., 2008]. High attrition (or low retention) rates are often experienced in computer science schools [Bennedsen and Caspersen, 2007], in part due to their students’ inability to learn programming, a core survival skill [Oman et al., 1989].

In America, the Computing Research Association’s statistics showed that the number of incoming undergraduates who list computer science as a probable major has fallen by 70% between the year 2000 to 2005 [Paul, 2007; Vegso, 2008]. It is reported that more than half of computer science majors change their major prior to graduation [Suter, 1996]. Furthermore, the majority of computer science students quit by the end of their freshman (beginner) year [Seymour and Hewitt, 1997].

In response to these scenarios, research in computer science education grows and is largely focused on aiding and improving teaching and learning of programming [Simon et al., 2008]. Sheard et al. categorised the themes of the papers presented in six computing education conferences. Their analysis stated that the themes of “ability”, “aptitude” and “understanding” have been researched in a total of 40% of all the accepted papers in these computing conferences. Second in the rank was the theme of “teaching and learning” and “assessment techniques” with 35% [Sheard et al., 2009]. Most of the papers in the Sheard et al. categorisations that fell into the “teaching and learning” and “assessment techniques” categories were taking into consideration the programming languages, different types of learning environments, introduction of assessment tasks such as Parson’s puzzles [Parsons and Haden, 2006], which we use in one of our surveys (we will explain in Chapter 6), peer assessments and tutoring. This shows that there are ongoing concerns about assessment as a tool to aid and improve teaching and learning. It also highlights the great attention to studies that seek to understand and strengthen students’ abilities to learn programming.

Specifically, a survey of the literature on the teaching of introductory programming in the past 30 years highlights that much of the researches that focus in computer science edu-
cation is on suitable languages for novice programmers and on the development of tools and environments to enhance learning and teaching pedagogies for programming courses [Pears et al., 2007]. These are evidence that teaching methods and learning tools are important for the improvement of learning programming.

Our own efforts to understand the difficulties of teaching and learning programming are motivated by novices’ performances, based on assessments of a particular semester. Novices’ performances outcomes may vary markedly. Figure 2.1 depicts novices’ performances based on assessments of 141 students in a Programming 1 course during 2009, which covered basic knowledge of the Java programming language. Table 2.1 explains the grading for the course based on assessment guides from RMIT University (source from: http://www.rmit.edu.au/students/gradingbasis/gpa).

Referring to Figure 2.1, at one end of the spectrum almost 35% of novices pass the course with flying colours, by obtaining “high distinction”. However, approximately 30% of the students “fail” the course, which causes them to repeat and lag in progressing to the next programming course. We understand that individuals respond differently to teaching techniques and we strongly believe that supporting current learning methods with learning taxonomies may be the best aid in minimising the gap between the two distinct spectra (those who obtain “high distinction” and “fail” grades) of students. In Chapter 3 we will discuss these taxonomies that classify the levels of cognition in learning.
CHAPTER 2. LEARNING TO PROGRAM

Table 2.1: Descriptions for Table 2.1

<table>
<thead>
<tr>
<th>Key code</th>
<th>Grade definition</th>
<th>Mark range</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>High Distinction</td>
<td>80-100</td>
</tr>
<tr>
<td>DI</td>
<td>Distinction</td>
<td>70-79</td>
</tr>
<tr>
<td>CR</td>
<td>Credit</td>
<td>60-69</td>
</tr>
<tr>
<td>PA</td>
<td>Pass</td>
<td>50-59</td>
</tr>
<tr>
<td>NN</td>
<td>Fail</td>
<td>0-49</td>
</tr>
<tr>
<td>WDR</td>
<td>Withdrawn from course</td>
<td>None</td>
</tr>
</tbody>
</table>

In related research, Berglund and Lister explain that the relationship between teachers and students are often neglected in computer education research. In physics courses, researchers are considering studying what, who and how they are currently teaching, as students claim that the courses are hard to learn [Lasry et al., 2009]. While several factors influence student performance outcomes, the interaction between instructors and students (the who-factor) through the classrooms and out-of-class engagements are the key determinant of how the students will fare. The course contents should be well-designed (the what-factor) and the method of delivery (the how-factor) should support the variability of students learning styles and abilities.

We use the Didactic Triangle [Kansanen, 1999; Kansanen and Meri, 1999; Kansanen, 2003] to explain the relationship between student, teacher and content in programming courses. The Didactic Triangle is a model used to describe teaching and learning situations. The three main entities to achieve didactic understanding are: teacher, student and content (Refer to Figure 2.2). The teacher, or instructor, guides the student through the learning materials (contents) and the student depends on such guidance to facilitate their learning. The learning contents are accessible by both instructor and student. Students are able to interact with instructors by responding to the instructor’s deliverables.

Therefore, in order to identify the learning difficulties faced by novices, we classify the challenges into three main categories based on Figure 2.2: **Student** - Novice Programmer Learning Complexities (refer Section 2.2.1); **Teacher** - Instructor Teaching Approaches (refer Section 2.2.2) and **Content** - Nature of the Programming Content (refer Section 2.2.3) and discuss these details in the following subsections.
2.2.1 Novice Programmers Learning Complexities

Tertiary computer science students are expected to learn programming as early as their first semester and typically continue to learn programming in subsequent semesters until they graduate. Programming courses have been widely reported as being challenging for novices, especially at the very beginning. Novices face programming difficulties within the first few weeks of the formal study period. Novices often despair and get de-motivated easily due to the frustration of being unable to produce a computer program that works successfully according to stated objectives [Carbone et al., 2009].

If a poor start is made by a novice in the formative stages, especially during the early days and weeks of learning programming, it often leads to ongoing difficulties with programming classes in the subsequent stages of study. Students enrolled in computer science courses typically encounter their early challenges when learning programming [Lahtinen et al., 2005; Milne and Rowe, 2002], especially for the first time. Some may succeed in their initial course, but may carry other conceptual “baggage” that hinders their understanding of concepts learned in further study.
Novice programmers are often known to have difficulty in grasping the foundation level programming concepts early enough, resulting in grief and frustration, and ultimately, surrender. Yet, those who do manage to overcome their learning difficulties are able to move on, even excel. Advice from students who have undertaken a basic programming course previously, to novices, usually includes encouragement to build strong foundations and places emphasis on learning the syntax in the early weeks, because if one falls behind in a programming class, it will be difficult to catch up [Hanks et al., 2009].

Incorrect models of a range of fundamental programming concepts are conveyed in teaching or received (by novices), often compounding the problems of understanding. Poor models of programming concepts have a propagating effect, plunging novices into a spiral of frustration and total loss of confidence and self-belief as more complex material is covered [Gray et al., 1993; Caspersen and Bennedsen, 2007].

Many researchers have written about the difficulties faced by novice programmers [Garner et al., 2005; Ginat et al., 2004; Kim and Lerch, 1997; Lister and Leaney, 2003; McCracken et al., 2001; Miliszewska and Tan, 2007]. Many novices think that programming is difficult because it requires a vast amount of knowledge and many skills to be learned all at once and at the very beginning [Spohrer and Soloway, 1986]. Additionally, novices face difficulty in putting the pieces together to compose a valid program [Winslow, 1996]. They also often do not examine the task given by instructors carefully enough [Ginat et al., 2004]. This may lead to erroneous programs as they do not meet the program specification. Furthermore, Ng et al. show that novices approach computer programming by taking “small steps” rather than looking at it as a whole [Ng et al., 2006].

Dehnadi conducted a study to understand novices, with a view to developing predictors of success in programming courses. He claimed that students who succeed in programming have an innate aptitude for programming. Furthermore, Dehnadi stressed that failures were attributed to lack of aptitude and never attributed to poor or inadequate teaching and assessment materials. We do not agree with this as we believe that anyone may succeed in learning programming provided they are given proper learning materials and support. Indeed, in follow-up research papers, they realized that the test they initially developed as an admission filter to avoid high failure rates in programming courses does not work [Bornt et al., 2008; Dehnadi, 2006; Dehnadi and Bornt, 2006]. We strongly believe novices need a proper learning aid to boost their capabilities to build a strong foundation in mastering the programming course’s contents.

In the following section, we identify a second factor that contributes to the challenges of
learning programming based on the teaching approach employed by the instructors.

2.2.2 Instructor Teaching Approaches

Research into novice programming difficulties have focused solely on students’ misunderstandings and misinterpretations of programming concepts. Indeed, the students’ perspectives are critical, if novice programming difficulties are to be alleviated.

As in Shuhidan et al., to our knowledge, there is no previous research in computer science education area has studied instructors’ perspectives of how and why student outcomes are measured in the ways that they are commonly measured, in the programming courses [Shuhidan et al., 2010]. Instructors design course materials and assessment tasks, with the latter perceived (by the instructors) as being suitable to test student understanding and ability to apply concepts learned, supported by classroom and out-of-class interactions. From the instructor’s perspective the assessments ostensibly provide accurate evaluations of student performance. The instructor, after all, plays an important role in initiating the steps for better learning.

Programming courses are not only hard to learn but also hard to teach [Berglund and Lister, 2010]. Most novices learn to program via formal instruction in a computer science introductory course (or frequently referred to as “CS1”) [Robins et al., 2003]. Guzdial stresses that teaching programming by expecting students to program is ineffective [Guzdial, 2009]. In additional, Guzdial and Robertson highlight that instructors expect novices to learn programming by constructing a program using basic information of the programming language and this is the same way as how expert programmers will do to learn a new programming language [Guzdial and Robertson, 2010]. Indeed, traditional ways of learning to program are like taking “one to the ocean” and expecting them to swim at once. Minimal guidance will not support learners, as learning programming is known to be a complex process (further discussions in Section 2.2.3).

Programming course instructors may teach and approach the students based on their own previous learning experiences [Lister, 2008]. We can assume that all programming instructors enjoyed their learning of the programming courses and succeeded in the courses with flying colours to qualify themselves to teach programming. Hence, based on their success, instructors have the tendency to introduce the students to their mode of learning, which may not be a suitable approach to all the students.

Furthermore, Lister stressed that instructors are accomplished programmers, hence they
provide reasoning in a form that may not be processed by the novices [Lister, 2011]. Thus, the delivery of the programming concepts may not be received by novices. In other research, Lister emphasises that programming instructors may assume that one mode of teaching may fit all learners. Variability of humans may not allow all students to easily adapt to one specific mode of learning. Instructors must allow flexibility, as different students learn at different paces and with different amounts at one time [Lister, 2008].

Clear et al. point out that two major contributors to the exodus from information technology degrees are poor teaching of computing in high schools and the “nerdy” image of the profession to young people [Clear et al., 2008]. We contend that, even at university level, too much is taken for granted by instructors in the formative stages of learning programming. Most instructors have come from the group that excelled in programming courses and they hardly faced any difficulties in learning to program computers. Therefore, some instructors teach programming based on how they relate to the course, and not how the students relate to it [Berglund and Lister, 2010].

Figure 2.3: The Task Typology [Carbone, 2007]

Carbone proposed eight dimensions of task typology to evaluate programming tasks given to students as depicted in Figure 2.3. The term typology is defined as the study of types of tasks based on specific characteristics. This typology aims to describe the characteristic of a task, in particular a programming task.
The eight dimensions of typology include: Routine - Novel dimension; Closed - Open dimension; Artificial - Authentic dimension; Degree of Ownership dimension; Degree of Linkage dimension; Degree of Reflection dimension; Individual - Collaborative dimension and Simple - Complex dimension. Each petal of the typology represents the characteristic to be evaluated. In the analysis of tasks using this task typology, Carbone mentions that most tasks are given to students resulting in lack of ownership and this will not allow students to explore learning. Furthermore, if a task has a high degree of difficulty in all dimensions, it may demotivate the students and they may possibly quit trying to solve the task. In a beginner course, the task or exercises given to students should not be too complex as to turn them away, but an increasing level of difficulty may help students to stay motivated in learning.

In the next section, we discuss the challenges to learn programming due to the nature of the course’s content.

2.2.3 Nature of the Programming Course’s Contents

Two of the critical aspects for teaching and learning programming highlighted by Schulte et al. were the domain knowledge for comprehending programs seems to be underestimated in pedagogy, and there are a lot of possible learning tasks for reading and comprehending programs [Schulte et al., 2010]. Therefore, decomposing task for learning programming is crucial to break the task into smaller components for ease of comprehending.

Lahtinen et al. states that the problem of learning programming is due to the intrinsic nature of the course, lack of resources and lack of personal instruction. Introductory programming textbooks and modules are mostly “knowledge-driven”; which emphasise knowledge of a particular language based on examples and exercises [Robins et al., 2003]. In related research, Thompson suggests textbook authors of object-oriented programming books should reduce the amount of material to ensure critical aspects become visible [Thompson, 2008]. Therefore it is important to design and organise the learning materials systematically to ensure the learning goals are met.

The steep programming learning curve demands the learning of the peculiarities of language syntax, being simultaneously creative and a good problem solver and having the patience to test and debug the program once it has been written. Rogalski and Samurcay stress that acquiring and developing knowledge about programming is a highly complex process [Rogalski and Samurcay, 1990]. Programming courses require the learner to acquire multiple skills simultaneously. Programming languages are like many spoken languages (also
refer to natural languages), they have a grammar (syntax) and descriptions of level of interpretation of the grammar (semantics).

Learning to program is similar in difficulty to the effort required for learning a new language [Parsons and Haden, 2006]. As an example, in English, we cannot sensibly learn how to use the verb “to be” without learning an entire complex set of syntactic and semantic relations, and this must be learned simultaneously [Sweller and Chandler, 1994]. This results in a high cognitive load in the early stages, for a novice to obtain the required skills. In this research, we consider it important to include both the syntactic and semantic knowledge tested in exam questions. It is interesting to investigate statistically the extent to which the syntax and semantic components are dependent on each other when instructors devise examination questions (further discussions in Chapter 6).

In a recent paper, an analysis of 11 sets of CS1 exam papers found that there were 39.0% of the questions covered about Object Oriented concepts and 45% of the questions analysed require the students to write code [Sheard et al., 2011]. However, it was not mentioned how the students response to the questions, whether the questions were easy, moderate or difficult. In other research, Milne and Rowe found that topics on pointers and memory-related concepts (such as copy constructors and virtual functions) proved to be the most difficult because the learners are incapable of creating clear mental models of their execution [Milne and Rowe, 2002].

Debugging is one of the skills required to program. Novices find it is difficult to debug a program. In Carbone’s thesis, one student confessed his difficulty in debugging as quoted below.

“Debugging was a nightmare. There were masses of code splashed all over the screen, all of which had begun to look the same. Loops which I thought had made sense initially no longer did and I realised I was in troubled waters” [Carbone, 2007].

In related research it is mentioned that novices frequently add bugs to the program while trying to fix a program that has errors [Gugerty and Olson, 1986]. Novices debug a program because “it just does not look right” [Murphy et al., 2008]. Furthermore, novices are expected to have problem solving skills in order to be a good programmer.

Above all, learning to program requires students to practise multiple skills simultaneously. Instructors may present the learning contents in various ways to ensure students are able to
CHAPTER 2. LEARNING TO PROGRAM

engage in learning. In the next section, we discuss the available learning aids that assist novices to adapt within a programming course.

2.3 Learning Programming Aids

Gómez-Albarrán reviewed tools and learning approaches that support learning or teaching programming. He categorised the tools into four groups; tools with a reduced development environment; example-based environments; tools based on visualisation and the simulation environments. Based on this review, we realised that there are many tools developed to aid programmers to program. Tools with reduced environments, or that involve simple development environments, that were suggested to suit novices are BlueJ (available at: http://www.bluej.org/) [Barnes and Kölling, 2008] and DrJava (available at: http://dr-java.org/), to name a couple. In the example based environments, Javy [Gómez-Martín et al., 2003] was developed with a set of exercises to aid teaching Java compilation with the help of a metaphorical virtual environment that simulates the Java Virtual Machine. However, in an evaluation of Alice, BlueJ, Jeliot 2000, Lego Mindstorms with Ada and RAPTOR, Gross and Powers emphasised that users of these tools tend to approach the program with an outcome-based rather than a process-based method [Gross and Powers, 2005]. This means that users focus on getting the correct outputs and care less about the process of how the program works.

2.3.1 Learning Through Exercises

PeerWise (available at: peerwise.cs.auckland.ac.nz/) is a repository of multiple choice questions contributed by learners that focuses on using sets of exercises to aid students learning to program. Students post programming questions to test their classmates or peers. The peers may attempt to answer and rate the difficulty of the questions [Denny et al., 2008c]. In related research, Denny et al. found that there exists a significant correlation between students’ overall performances in exams and their contributions to the multiple choice questions in Peerwise [Denny et al., 2008a]. In other words, the students who perform best in the course, contributed more multiple choice questions compared to the others.

Both Ceilidh [Benford et al., 1993] and its new version, CourseMarker (available at: http://www.cs.nott.ac.uk/~cmp/cm_com/index.html) focus on producing a platform that provides prompt feedback and assessment of text-based or multiple choice answers [Higgins et al., 2005]. CourseMarker has expanded its capacity to automatically mark other courses,
not only programming courses [Higgins et al., 2003]. The development of CourseMarker is intended for automatically marking assessments, and not into classifying the questions to be tested.

In related research, Ville, (available at: ville.cs.utu.fi/) was found to be similar to CourseMarker. Ville is a platform to automatically assess exercises and in addition, Ville gathers data about students’ learning behaviour [Kaila et al., 2008].

A learning platform called Trakla (available at: http://www.cs.hut.fi/tred/WWW-TRAKLA/WWW-TRAKLA.html) is produced to distribute exercises and evaluate answers automatically. It was originally designed to assess the data structures and algorithms courses at Helsinki University of Technology [Hyvönen and Malmi, 1993]. In the improved version, Trakla2 (available at: https://trakla.cs.hut.fi/) provides a positive effect on learning of programming [Malmi and Korhonen, 2004]. Korhonen and Malmi also introduced learning taxonomies to classify algorithm simulation exercises to support personalised learning using Trakla and Trakla2 [Korhonen and Malmi, 2004].

The Web-based Inquiry Science Environment (WISE) (available at: http://wise.berkeley.edu/) system emphasises learning instruction using case studies [Linn and Clancy, 1992; Clancy and Linn, 1992] to help novices to learn. In an evaluation of WISE, the students responded that the programming course was enjoyable and the role of the instructor had changed fundamentally. Clancy et al.’s research has shown that tutoring has been more effective in terms of engaging with the programming students compared to engaging with the primary lecturer [Clancy et al., 2003]. A small group of students in a tutorial may benefit from the attention from their tutors to cater to the different needs in learning programming as compared to a large number of students in a lecture.

2.3.2 Visualisation may Boost Learning

Ville was introduced as an assessment tool and also a visualisation tool that supported a “language-independent” platform, meaning it may be used for Java language, C++ language or for a language that a user designs on their own [Rajala et al., 2007]. Also, the Multi-User Programming Pedagogy for Enhancing Traditional Study (MUPPETS) encourages interactions in interactive learning in a virtual environment [Phelps et al., 2003].

Alice (available at: http://www.alice.org/) uses a visualisation approach to encourage learning programming within a 3D animation environment [Pausch et al., 1995; User Interface Group, 1995; Conway, 1997; Cooper et al., 2000]. In a small scale evaluation, Moskal
et al. found that “at risk students” that participated in Alice, on average, received significantly higher grades than “at risk students” who did not participate in Alice [Moskal et al., 2004]. They also emphasise that Alice is a platform for learners to familiarise themselves with computer programming before they progress to learning to read and write using the “real” programming language. However, Powers et al. stress that Alice may boost confidence for novices to program computers but causes damage in the transition to the next programming course [Powers et al., 2007].

2.3.3 Discussion on Learning Tools

Programming is a practical course, therefore, learning through exercises is a good approach to encourage and engage novices to learn programming. The existing tools, as listed above, emphasise exposing novices to interactive learning to attract and keep them motivated to learn. As an example, in Alice, novices are encouraged to practice their logic skills by using the existing functions to create an animated output. Another example is Peerwise that support learning with peers where students share questions among themselves and discuss them in a forum. We found that these tools have a lack of research in classifications and in the scaffolded levels of exercises given to the students. The classifications are important in formative assessment as students may progress from one scale level to another and need to be able to identify and keep track of their own levels of learning. On the other hand, as in summative assessment form, instructors can make use of the multiple levels of exercises to test their students based on the learning objectives. The discussion on the assessments will be provided in the next section.

2.4 Study of Assessment

In the previous section, we addressed novice challenges to learning programming in the context of a traditional delivery model of teaching and learning. This model is still widespread and uses lectures, tutorials and laboratories as well as assessments to teach programming and assess a student’s ability.

The course materials are lecture notes, tutorials and laboratory exercises. Lectures are delivered to a large number of students each week, guided by the lecture notes distributed to students. For tutorial and laboratory sessions, students form small groups, typically allocated randomly. In a tutorial, students often discuss the theories of programming, and in the laboratory class, students are expected to practise programming concepts. Both of these
contexts increase the level of interactions between students and instructors over and above that possible in lectures.

A student’s ability to learn is often measured by assessments. Educational sectors continue to require high standards from their students [Swanson, 1991]. Hence, assessment is structured into formative and summative assessments, and typically both contain hurdles. The two types of assessments often, but not always, contribute to the final grading of a student. Therefore, success is usually measured by the marks gained in these assessments.

Assessment is structured into formative and summative assessments and typically both are used as hurdle tasks. Formative assessments may assist students to learn and prepare themselves to sit for summative assessment. Therefore, it is crucial to select suitable exercises to aid students in overcoming learning difficulties in reading and writing computer programs. The next subsections will explain the formative and summative assessments.

2.4.1 Formative Assessment

Formative assessment is meant to provide practice to students over a period of study. This is important to guide the students on what needs to be learned based on the learning outcome of the course to ensure that they are on the right track. Formative assessment is part of the instructional process and the key to student’s achievements [Garrison and Ehringhaus, 2007]. Furthermore, Garrison and Ehringhaus emphasised that formative assessment allows for interventions and adjustments to learning instructions while the course is still running. If students provide wrong answers in the formative assessments, it shows that they have misconceptions or do not understand the particular topic. Both students and instructors may benefit from these interventions. Instructors may respond promptly to the students if they show signs of having problems and students may benefit by being on track as they progress throughout the learning period. Early corrections of misconceptions may save the students from failing.

Harlen and James suggested the characteristics of formative assessment are as:

- it is essentially positive in intent, in that it is directed towards promoting learning; it is therefore part of teaching;
- it takes into account the progress of each individual, the effort put in and other aspects of learning which may be unspecified in the curriculum; in other words, it is not purely criterion-referenced;
CHAPTER 2. LEARNING TO PROGRAM

• it has to take into account several instances in which certain skills and ideas are used and there will be inconsistencies as well as patterns in behaviour; such inconsistencies would be “error” in summative evaluation, but in formative evaluation they provide diagnostic information;
• validity and usefulness are paramount in formative assessment and should take precedence over concerns for reliability;
• even more than assessment for other purposes, formative assessment requires that pupils have a central part in it; pupils have to be active in their own learning (teachers cannot learn for them) and unless they come to understand their strengths and weaknesses, and how they might deal with them, they will not make progress [Harlen and James, 1997]

In later chapter, we would like to propose a formative assessment to promote learning in programming, so that students may self assess their performance.

2.4.2 Summative Assessment

Summative assessment tests students periodically at the end of a particular teaching and learning period, to discover where the students’ abilities lie with respect to the learning outcome [Garrison and Ehringhaus, 2007]. It can happen every few weeks, months or once a year. It is also conducted to assess the effectiveness of a program, learning goals or to determine a student’s placement in a course.

Harlen and James also suggested the characteristics of summative assessment are as;

• it takes place at certain intervals when achievement has to be reported;
• it relates to progression in learning against public criteria;
• the results for different pupils may be combined for various purposes because they are based on the same criteria;
• it requires methods which are as reliable as possible without endangering validity;
• it involves some quality assurance procedures;
• it should be based on evidence from the full range of performance relevant to the criteria being used [Harlen and James, 1997]
We plan to study novice and instructor responses to questions tested in the summative assessments in the hope of developing a formative assessment, which may help novice programmers learn and guide them to think of what they “should be thinking” and what “should be highlighted” in order to attain the fundamentals of programming. The proposed formative assessment may also help to minimise the misconceptions of fundamental programming concepts. We plan to develop the formative assessment using multiple choice questions as the one of the test instruments.

2.4.3 Multiple Choice Questions

In this section, we will discuss the multiple choice questions as test instruments for programming assessments.

“Multiple choice items are, at least in their average and widely used forms, exercises in detection and selection rather than generation. They often enforce a view of single correct answers at the expense of recognizing culturally variegated forms of excellence or contrasting approaches to displaying understanding” [Dennie Wolf and Gardner, 1991].

Multiple choice questions are found to be useful to test the students’ knowledge and to eliminate the “learn by heart” stereotype of students who do not understand the course content [Marshall, 2008]. Multiple choice questions have been reported to be deceptively easy to write, but creating a good set of multiple-choice questions is actually quite difficult to attain [Wood, 2003]. Multiple choice questions, as a formative assessment, have been shown to attract students to participate in learning [Ramesh et al., 2005; Denny et al., 2008a].

Multiple choice questions should not be seen as being too easy in the exam, since one third to half of a Programming 1 class failed to achieve the 70% pass figure on their first attempt [Lister and Leaney, 2003]. Indeed, the authors believe that multiple choice questions can provide a solid test of a student’s knowledge and comprehension [Lister and Leaney, 2003]. We recognise the high value of multiple choice questions because it may aid novices to learn if applied in formative assessment.

Scouller conducted a survey to examine the relationship between the learning approach and the assessment method, with regards to multiple choice questions in final examinations and essays in assignment [Scouller, 2004]. Their results showed that students were more
likely to employ surface learning approaches in the multiple choice question examination context and to perceive multiple choice question examinations as assessing knowledge-based (low levels of) intellectual processing. However, poorer performance in the multiple choice questions examination was associated with the employment of deep learning strategies, and may be due to deep learners tending to think more than is required to solve the multiple-choice questions. In contrast, students were more likely to employ deep learning approaches when preparing their assignment essays which they perceived as assessing higher levels of cognitive processing.

More recently, there have been efforts to set up a bank of exam questions. The BRACElet [Lister et al., 2006] group promotes a set of questions to be tested in exams, such as SOLO thinking-out-loud questions [Lister et al., 2004], Parson’s puzzle questions [Parsons and Haden, 2006], Donna’s marbles questions [Fidge and Teague, 2009]. These efforts are highly regarded and the new approaches to questions have been exposed to students all over the world. It is in our interest to know the instructors’ views of these types of questions and we will discuss our research around this area in Chapter 6.

2.5 Research of Programming Errors

In this research, we are primarily interested in ensuring that novices have a good foundation to build upon in programming, to ensure that further study involving programming is well-received. In a review of literature in debugging from the educational perspectives, McCauley et al. mentioned that programming errors may result from misconceptions held by programmers [McCauley et al., 2008]. Improved understanding of novice errors will also better inform educators about alleviating the difficulties experienced by novices at commencement.

In an analysis of 108,652 records of student errors in programming for a semester of study showed that 36% of the errors are syntax errors, 63% are semantic errors, while 1% of them are lexical errors [Ahmadzadeh et al., 2005]. Semantic errors are errors in design or composition of programs. Therefore, it is crucial to scaffold learning as novices may find that it is hard to decompose learning and translate problems into design and then to write a program as the solution. Syntax errors, although they are significant to novices [Kummerfeld and Kay, 2003; Vee et al., 2006], can be fixed through development environment (Refer Section 2.3).
CHAPTER 2. LEARNING TO PROGRAM

The most common syntax errors are wrong “type” in declaring a variable or argument or assigning to a variable of the wrong type [Vee et al., 2006].

In other research regarding bug categorisation, seven components that novices tend to misunderstand were identified as input, output, initialisation, update, guard, syntax and plan [Spohrer et al., 1985].

Also, Garner et al. [Garner et al., 2005] reported that the top three common errors for novices are basic mechanics (which involve syntax errors in basic program structures including semicolons and curly braces as an example), having issues with program design and having problems with basic structure.

Students, in particular novices, put less effort into program design, which can result in buggy programs. They tend to not carefully examine the required task given. Buggy programs reveal misconceptions about a particular topic [Ginat et al., 2004]. Bugs may arise as a result of planned decomposition problems and also due to difficulties in piecing together separate parts of the program. These misconceptions may be one of the causes of learning difficulties [Hammer, 2003]. In addition, novices overrate their skills for debugging programs [Sheikh Aljunid and Shuhidan, 2006]. Their perception is highly rated compared to their ability to debug programs.

Essentially, the highest error rates occur in the first three weeks of a semester, which usually runs over 12 weeks, based on the analysis of a collection of students’ programming errors [Marceau et al., 2011]. We name this as the “critical-period”, as novices have to gain various skills at once, which is one of the main issues we highlight in this thesis.

Kopec et al. analysed programmers’ examination errors, but focused on intermediate programmers. Intermediate programmers are those who have some programming experience and understand basic concepts of programming [Kopec et al., 2007]. Incorrect planning leads to incorrect answers. They concluded that educators must be very careful in their problem description and presentation, and that novices are often confused with nested loops and recursion, which differentiate intermediate to novice programmers’ errors.

In an evaluation of a model of programming errors using Alice, novice programmers spent an average of 50% of their time debugging errors that involved knowledge and understanding implementation artefacts, implementing and modifying algorithm, language constructs and the usage of libraries. These errors lead to further programming errors [Ko and Myers, 2003].
2.6 Summary

This chapter highlights the fact that novices do face real difficulties in learning to program. We classify the learning difficulties according to the Didactic Triangle [Kansanen, 1999] that is based on three categories: teacher, student and content. Fundamental understanding of programming concepts taught in early weeks is crucial to building up programming skills.

Assessments play an important role in encouraging learners to explore programming. Furthermore, the instructor’s choice of suitable assessments to support the learners’ capabilities to program computers is essential in determining the true success or failure of learners. In developing an assessment, instructors creatively design the questions and carefully choose the suitable test instruments to portray that novices have meet the learning objectives. Multiple choice questions may be a reliable test instrument that support learning programming in early stages. Hence we will study multiple choice questions in summative assessment in Chapter 5.

There are some existing tools developed to support learning programming as in the form of formative assessment. Researchers focus on learning through exercises and visualisation to aid novices to learn programming. Our aim is to develop a guided learning tool, but it cannot be just another tool to aid novices to learn programming. Therefore, in the next chapter (Chapter 3), we explore the theories of learning that we may use to facilitate novice learning of programming. We realise that there is a need for better instructions and to organise course content to scaffold early learning of programming, especially for novices. These two factors are found to be lacking in existing learning tools and will be implemented in our proposed guided learning tool.
Chapter 3

Cognitive Theories of Learning

The key challenge to learning programming is to acquire many skills at the same time (Refer to Section 2.2.1). Novices have to actively juggle the language syntax and semantics as well as grapple with problem solving skills. Furthermore, programming courses require one to study programming theoretically and be able to apply practical work to design and implement programs. Therefore, this situation pushes novices to learn different forms of knowledge and practise skills at the same time, and this may cause their cognitive processes to overload. The process of transferring information in the human brain is widely known in psychology as the cognitive process. It involves the capacity of the brain to process information.

This chapter will cover the discussion surrounding the cognitive processes (Section 3.1) and learning taxonomies in relation to teaching and learning of computer programming. We focus on the Bloom (Section 3.2) and SOLO (Section 3.3) taxonomies to classify questions and novice responses in summative assessments or final examinations. Bloom’s Taxonomy was introduced to classify thinking levels in general education whereas the SOLO Taxonomy was developed to classify the overall thinking processes. In relation to the learning taxonomies, we discuss constructivist theory (Section 3.4) and learning styles (Section 3.5) that may influence novice success in learning programming.
CHAPTER 3. COGNITIVE THEORIES OF LEARNING

3.1 Cognitive Process

In relation to the cognitive processes, Yousoof et al. stated that humans have limited working memory to process complex knowledge to form long term memory [Yousoof et al., 2005]. Novices may process information in seven chunks, plus or minus two chunks, at any one time [Miller, 1955]. In more recent research, Cowan emphasised that the short term working memory is limited to four, not seven plus or minus two chunks at a time [Cowan, 2000]. As discussed in Chapter 1, *Hello World* program, there are at least 14 new terminologies for novices to remember, understand and apply to write a similar or more advanced program. Clearly, this may cause a cognitive overload for some novices because this is far too much information to be processed at one time.

One way to reduce cognitive process overload is through frequent and repetitive practise with small chunks of information [Clark and Taylor, 1994]. These exercises may help to ease the transfer of information from working memory to the long term memory. Frequent repetition of “familiar” chunks will allow the brain to focus on the “unfamiliar” chunks, and will therefore improve learning capacity. The well known limitations of working memory do not apply to all data, but only to data that is not already in long term memory [Lister, 2008].

Learning taxonomies like Bloom and SOLO are also useful to help novices to scaffold cognitive processes. Learning taxonomies may extract learning to multiple levels of tasks. Therefore, the learning complexity can be reduced by introducing students to a low level task and then allowing them to progress to tasks of a higher level of complexity.

The main problem when learning to program is not a lack of understanding the basic concepts but rather learning to apply them [Lahtinen et al., 2005]. In fact, novices may be able to understand the concepts but find it difficult to implement them [Butler and Morgan, 2007]. The inability of novices to produce a clear mental model may also cause them difficulty when learning programming [Milne and Rowe, 2002]. In an investigation of perceptions of program correctness, more than half of the responses in the study did not reach the mature stage of understanding what it means to program [Stamouli and Huggard, 2006]. Therefore, understanding or applying concepts may be the reason for the learning difficulty. We see the need to scaffold learning using learning taxonomies to aid novices to achieve their learning objectives in certain stages or levels. We will discuss learning taxonomies and apply them to the learning of programming in the next section.
3.2 Bloom’s Taxonomy

Bloom’s Taxonomy categorises levels of learning by highlighting the involvement of cognitive processes required in solving a particular question. In this research, we study and describe both the original and revised versions of Bloom’s Taxonomy and compare them.

3.2.1 Original Bloom’s Taxonomy

Benjamin Bloom created a taxonomy of thinking levels in the 1950s, known widely as *Bloom’s Taxonomy* [Bloom and Krathwohl, 1956]. Bloom’s Taxonomy aims to motivate educators to focus on a holistic model of education. The model has three main domains of learning outcomes referred to as: affective, psycho-motor and cognitive. The affective domain involves attitude, emotion and feelings towards learning. The psycho-motor domain focuses on the physical movements as a manipulative tool to attain a goal. Last but not least, the cognitive domain supports the process of thinking to attain knowledge.

As mentioned in Section 3.1, we aim to minimise novices’ cognitive loads through repetitive practise and scaffolded tasks. These involve the cognitive domain of Bloom’s Taxonomy. Therefore, the rest of this section will discuss the thinking levels of the cognitive domain.

Educators use Bloom’s Taxonomy to guide the composition of challenging and sophisticated activities for learning. The cognitive domain of study remains relevant today and is employed in identifying the educational levels required for course outcomes. Bloom’s Taxonomy presents six levels of thinking or cognitive domains: knowledge, comprehension, application, analysis, synthesis and evaluation (Refer to Figure 3.1).

In terms of cognitive complexity, knowledge is the lowest level and relates to memorising information and being able to recall definitions. The second category, comprehension, involves the ability to make use of the memorised information. The third level, application, requires the ability to apply information without a given guide. The fourth level, analysis, requires the ability to discuss the relationship between subcomponents of related information. The fifth level, synthesis, involves arranging disorganised elements to produce well ordered form. Evaluation is the highest level of cognition and relates to the creating, developing, writing and critiquing of ideas and abstractions.

As an example, if one is able to memorise the definition of an “integer” as a data type, this is categorised in the knowledge level and the ability to describe its usage is in the comprehension level of Bloom’s Taxonomy. Next, if given a scenario where one has to write a program, if one is able to identify that an “integer” should be used without a given definition,
this is in the *application* level of Bloom’s Taxonomy. If a calculation involves multiple data types is presented to novices and they have the ability to distinguish between the usage of an “integer” and other data types, this is in the *analysis* level. Next, in the *synthesis* level, one is given a programming project and is required to propose a solution which includes the usage of different data types to produce an efficient result from the calculation. Finally, the *evaluation* level involves judging others’ programming work and providing critiques of it. Isaacs outlined the categories of each level of Bloom’s Taxonomy with descriptions and some example of keywords associated with them in Table 3.1 [Isaacs, 1996].

### 3.2.2 Revised Bloom’s Taxonomy

In 2001, Anderson et al. revised the major categories in Bloom’s Taxonomy to suit the emerging educational institutional needs of the new century. The revised Bloom’s Taxonomy maintained the original ideas of Bloom and Krathwohl, being the levels of cognition, but made changes within the categories, expanding them and explaining them in the context of general education. Furthermore, Anderson et al. claimed that the revised Bloom’s Taxonomy
**Table 3.1: Bloom’s Taxonomy and Descriptions (Sourced from [Isaacs, 1996] as referred to [Bloom and Krathwohl, 1956])**

<table>
<thead>
<tr>
<th>Category</th>
<th>Descriptions</th>
<th>Sample Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>Able to judge the value of ideas, works, solutions, materials or methods</td>
<td>Judge, appraise, evaluate, rate, compare, value, revise, score, and select</td>
</tr>
<tr>
<td>Synthesis</td>
<td>Able to pull together many disorganised elements or parts so as to form a coherent whole</td>
<td>Compose, plan, propose, design, formulate, arrange, assemble, collect, construct, choose, assess, estimate and measure</td>
</tr>
<tr>
<td>Analysis</td>
<td>Able to break down a communication into its constituent parts, revealing the relationships among them</td>
<td>Distinguish, analyse, differentiate, appraise, calculate, experiment, test, compare, contrast, create, design, set-up, organise and manage</td>
</tr>
<tr>
<td>Application</td>
<td>Able to apply appropriate abstraction without having guidance on how to use it in that situation</td>
<td>Translate, interpret, apply, employ, use, demonstrate, dramatise, practise, illustrate, criticise, diagram, inspect, debate, inventory, question, relate, solve, examine and prepare</td>
</tr>
<tr>
<td>Comprehension</td>
<td>Able to make use of the materials or idea given</td>
<td>Restate, discuss, describe, recognise, explain, express, identify, locate, report, operate, schedule, shop and sketch</td>
</tr>
<tr>
<td>Knowledge</td>
<td>Able to remember, recall ideas, material or phenomena</td>
<td>Know, define, memorise, repeat, record, list, recall, name, relate, review and tell</td>
</tr>
</tbody>
</table>

is a more comprehensive taxonomy compared to the original taxonomy.

Figure 3.2 illustrates the dimension of cognitive processes of the revised version of Bloom’s Taxonomy, which we have applied to the cognitive domain.

The lowest level, *remember* requires a learner to recall relevant knowledge from their long term memory. Next, the *understand* level requires a learner to construct meaning from the given information. In the *apply* level, a learner needs to carry out the procedure in
a particular specified situation. The higher levels, analyse, evaluate and create have been assigned to questions which are used to test novices in their assignments or projects. At the analyse level, learners are required to break programs into parts and determine the connection between the parts and further determine the overall structure. In evaluation, students are required to look at the work of others and make judgements about them. Finally, in the create level, students reorganise elements to form a new pattern or structure.

3.2.3 Comparison of Bloom’s Taxonomy [Bloom and Krathwohl, 1956] and Revised Bloom’s Taxonomy [Anderson et al., 2001]

The number of the categories of the original Bloom’s Taxonomy is the same as the revised Bloom’s Taxonomy, the most obvious changes is the change of the terminologies from noun to verb aspects. Knowledge was renamed as remembering as learner’s acquire knowledge through the ‘remembering’ skills. Comprehension was renamed as understanding. The other categories are similar except that the revised Bloom’s Taxonomy eliminates the synthesis category. Synthesis is found to be redundant, as is the analysis category. Instead, the revised Bloom’s Taxonomy introduces the create category which explains if one is able to develop a new invention, this will require the highest cognitive skill.
CHAPTER 3. COGNITIVE THEORIES OF LEARNING

The lowest level of the original Bloom’s Taxonomy, knowledge has the largest amount of space among the six categories. This skill is used the most in the early stages of learning particularly when memorising information. Hence, this is the largest component in the levels of Bloom’s Taxonomy. The original Bloom’s Taxonomy can be illustrated as a pyramid (refer Figure 3.1). The pyramid represents the amount of time the brain spends in each level. As the pyramid goes higher, higher cognitive skills are required, but they are required less often. As the pyramid goes higher, higher cognitive skills are required, but they are required less often. Plus, once the lower categories have been established, the information has been stored in the long term memory and once the learner has conquered the highest level of the taxonomy, it will be easy to practise the ability in lower levels. Therefore, the evaluation has the smallest amount of space due to the original Bloom’s Taxonomy’s emphasis learner on having a strong foundation in a particular course in order to succeed.

On the other hand, the revised Bloom’s Taxonomy is illustrated as an inverted pyramid (refer Figure 3.2) with the three upper levels placed next to each other horizontally. We explain this as: the higher the levels of the Bloom’s Taxonomy, the more cognitive processes required. Therefore, Figure 3.2 shows more space in the higher level of Bloom’s Taxonomy compared to the lower levels.

In Chapter 5 we apply the original Bloom’s Taxonomy and revised Bloom’s Taxonomy. Further discussions of the application of the Bloom Taxonomy are in Section 5.2 and Section 5.4.

In developing the guided learning tool (Chapter 7), we found that the revised Bloom’s Taxonomy explained the cognitive levels well. We decided that the lowest level in the original Bloom’s Taxonomy should not be named knowledge, but remember (as in the revised Bloom’s Taxonomy). The main reason is that the learner acquires knowledge through the process of remembering. Therefore, the category is better suited to having the name as a verb: remember, but not the end goal: knowledge.

In the next section we highlight research in computer science that has used Bloom’s Taxonomy to classify assessments and learning activities.

3.2.4 Application of Bloom’s Taxonomy to the Computer Science Area

Bloom’s Taxonomy has long been used in the education sector, but only as a hierarchical model for the cognitive domain in the general educational area [Bloom and Krathwohl, 1956]. In computer science education area, there are a number of studies that applied Bloom’s Tax-
CHAPTER 3. COGNITIVE THEORIES OF LEARNING

onomy to programming courses [Oliver et al., 2004; Whalley et al., 2006]. Scott explains some links between programming questions and Bloom’s Taxonomy. He demonstrates how Bloom’s Taxonomy works in programming tests and provides some sample questions in each category of Bloom’s Taxonomy [Scott, 2003]. Scott’s work is a good introductory piece of research that relates programming questions to levels of Bloom’s Taxonomy but does not present any data for evaluation. Consequently, we classify our own programming questions from a summative assessment based on Bloom’s Taxonomy and study the relationship between our classifications of Bloom’s Taxonomy, and instructors’ and novices’ responses. Further details are in Chapter 6.

Lister and Leaney identify weak, middle and strong programming students based on criterion-referenced grading (grades which were assigned according to specified criteria, irrespective of the resultant grade distribution) [Lister and Leaney, 2003]. There are different treatments applied in order to obtain achievable grades. These treatments are based on the levels of Bloom’s Taxonomy. They also propose a scale based on the students’ performances to determine their progression to the following semester. In a study across 12 institutions, researchers employed Bloom’s Taxonomy and grounded based analysis to identify the novices’ strategies to solve problems based on thinking aloud [Fitzgerald et al., 2005]. Bloom’s Taxonomy is also found to be beneficial to specify learning outcomes in computer science courses [Starr et al., 2008].

However, despite the benefits of Bloom’s Taxonomy, Fuller et al. claims that it is not appropriate to be used in practical courses such as programming [Fuller et al., 2007]. They suggest two dimensions of Bloom’s Taxonomy with subcategories of producing and interpreting. In our context, it is not necessary to use this version of the revised Bloom’s Taxonomy as we focus on the early weeks of learning content, which cover basic programming knowledge and we would not expect the students to demonstrate the higher levels of Bloom’s Taxonomy (producing and interpreting) in the learning tool. In related research, Winslow claimed that novices seem to fail in applying knowledge of relevant areas. Bloom’s Taxonomy may clarify Winslow’s research as novices may not be able to achieve the high levels of the taxonomy due to the limitation of their cognitive processes.

Another issue includes the difficulty of distinguishing between the levels and categories after the question has been written by instructors [Thompson et al., 2008]. This is due to the nature of programming assessments, since there are rarely any suitable keywords listed from Bloom’s Taxonomy. In classifying the programming questions, we found that the sample keywords listed may be unhelpful, but the descriptions in each of the levels are well-explained.
3.3 SOLO Taxonomy

Another learning taxonomy, the SOLO Taxonomy [Biggs and Collis, 1982], evaluates the interconnection of each learning component based on learners’ responses. SOLO is an acronym for Structure of the Observed Learning Outcome and was introduced by John B. Biggs and Kevin F. Collis, in 1982.

Table 3.2: SOLO Taxonomy

<table>
<thead>
<tr>
<th>Learning Level</th>
<th>Prestructural</th>
<th>Unistructural</th>
<th>Multistructural</th>
<th>Relational</th>
<th>Extended Abstract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>No understanding</td>
<td>Portray understanding of a component of relevant knowledge</td>
<td>Portray understanding of few components of relevant knowledge</td>
<td>Able to relate multiple components of relevant knowledge</td>
<td>Able to generalise set of knowledge to a new domain</td>
</tr>
<tr>
<td>Mental Representation</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
</tbody>
</table>

The SOLO Taxonomy provides a more qualitative way to classify cognitive processes [Biggs, 2008b]. There are five categories in the SOLO Taxonomy: prestructural, unistructural, multistructural, relational, extended abstract. We present the learning levels, descriptions and the mental representations in Table 3.2.

The lowest level is prestructural whereby a learner demonstrates no understanding of the topic. Next, in the unistructural level, the learner shows a little understanding of the topic. In the third category, multistructural, the learner indicates partial understanding, limited to several components of the topic. The two levels that show strong foundations are relational and extended abstract. In extended abstract, the learner explores beyond the teaching scope. A more comprehensive explanation of SOLO Taxonomy with sample keywords are explained.
in Table 3.3 [Biggs et al., 1999].

Table 3.3: SOLO Taxonomy [Biggs et al., 1999]

<table>
<thead>
<tr>
<th>Phase of learning</th>
<th>Levels of understanding</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualitative Phase</td>
<td>Extended Abstract : Student conceptualises at a level extending beyond what has been dealt with in the actual teaching. Can generalise to a new area.</td>
<td>Theorise Generalise Reflect Generate Hypothesise</td>
</tr>
<tr>
<td></td>
<td>Relational : Indicate orchestration between facts and theory, action and purpose. Understanding of several components which are integrated conceptually. Can apply the concept to familiar problems or work situations.</td>
<td>Compare Causes Contrast Integrate Analyse Relate Apply</td>
</tr>
<tr>
<td>Quantitative Phase</td>
<td>Multi-structural : Indicates understanding of boundaries but not of systems. Understanding of several components but the understanding of each is discreet. Disorganised collection of ideas or concepts around an issue. Has not been able to relate the items in the list.</td>
<td>Describe Combine Classify Enumerate Algorithms List</td>
</tr>
<tr>
<td></td>
<td>Uni-structural : Concrete, minimalist understanding of an area. Focuses on one conceptual issue in a complex case.</td>
<td>Identify Memorise Do simple procedure</td>
</tr>
<tr>
<td></td>
<td>Pre-structural : No understanding demonstrated.</td>
<td>Miss the point</td>
</tr>
</tbody>
</table>

However, as originally specified in the SOLO categories [Biggs and Collis, 1982], this taxonomy was developed based on the age of the learner; the lowest category for the youngest learners and the level of categories increase as the learners grow older. We argue that learning is a process and there will always be something new to acquire at different ages or stages of life. Hence, the levels of cognition can be applied to learning as a process, regardless of age.
3.3.1 Application of SOLO Taxonomy in Computer Science Area

There are a few studies which use the SOLO Taxonomy to evaluate the responses of novices learning to read programs [Thompson, 2007; Lister et al., 2006; Whalley et al., 2006]. In Whalley et al., novices were asked to describe a program in plain language. The guidelines they used to categorise the responses are outlined in Table 3.4.

Table 3.4: SOLO Taxonomy Applied to Learning of Programming [Whalley et al., 2006]

<table>
<thead>
<tr>
<th>Category</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>Provides a summary of what the code does in terms of the code’s purpose.</td>
</tr>
<tr>
<td>Multistructural</td>
<td>A line by line description is provided of all the code. Summary of individual statements may be included.</td>
</tr>
<tr>
<td>Unistructural</td>
<td>Provides a description for one portion of a computer program.</td>
</tr>
<tr>
<td>Prestructural</td>
<td>Substantially lacks knowledge of programming constructs or is unrelated to the question.</td>
</tr>
<tr>
<td>Blank</td>
<td>Question not answered</td>
</tr>
</tbody>
</table>

In our research, we use the SOLO Taxonomy to categorise novice written-programming responses. We look at each component in a program, to learn if novices are able to relate the components to each other to achieve the relational level of SOLO Taxonomy. Lister et al. classified written and think-aloud responses from respondents based on the SOLO Taxonomy. They reported that novices tend to understand code via a line-by-line approach, rather than understanding the code as a whole.

In other research, Lister et al. proposed SOLO categories specifically for code writing task (refer Table 3.5) [Lister et al., 2010]. The categories were explained to suit the analysis of a computer program. Whalley et al. conducted an advanced research of the categorisation proposed by Lister et al. and they mentioned that the educators need to be careful in applying SOLO Taxonomy to student responses [Whalley et al.]. Whalley et al. also stressed that in analysing the students’ responses, the SOLO categories can be misinterpreted to portray the level of abstraction required by the students.

3.4 Constructivist Theory

The constructivist theory of learning was introduced by Jean Piaget. It is a theory describing how learning happens, with each individual learning differently, in terms of pace and amount.
CHAPTER 3. COGNITIVE THEORIES OF LEARNING

Table 3.5: SOLO Categories for Code Writing Task [Lister et al., 2010]

<table>
<thead>
<tr>
<th>Phase</th>
<th>SOLO category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualitative</td>
<td>Extended Abstract –</td>
<td>Uses constructs and concepts beyond those required in the exercise to provide an improved solution</td>
</tr>
<tr>
<td></td>
<td>Extending [EA]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relational –</td>
<td>Provides a valid well structured program that removes all redundancy and has a clear logical structure. The specifications have been integrated to form a logical whole.</td>
</tr>
<tr>
<td></td>
<td>Encompassing [R]</td>
<td></td>
</tr>
<tr>
<td>Quantitative</td>
<td>Multistructural –</td>
<td>Represents a translation that is close to a direct translation. The code may have been reordered to make a valid solution.</td>
</tr>
<tr>
<td></td>
<td>Refinement [M]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unistructural –</td>
<td>Represents a direct translation of the specifications. The code will be in the sequence of the specifications.</td>
</tr>
<tr>
<td></td>
<td>Direct Translation [U]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prestructural [P]</td>
<td>Substantially lacks knowledge of programming constructs or is unrelated to the question.</td>
</tr>
</tbody>
</table>

of information processed [Wadsworth, 1996]. In response to the uniqueness of the learning style of each individual, constructivist theory supports learners by trying to make sense of the subject on their own, by developing their own mental model along with the teachers guidance and direction [Richardson, 1997].

Constructivist theory is learner-centric and encourages students to create their own models of knowledge. Constructivist teaching approaches are suited to Bloom’s Taxonomy [Marlowe and Page, 2005], which range from rote learning levels to deeper comprehension levels that allow more complex problems to be solved. In this theory, learners are free to explore using the multiple levels of the learning taxonomy.

In other research, Booth introduces “transmissive pedagogy” that emphasises the assumption made that students absorb the knowledge transferred from instructors and apply it [Booth, 2001]. Thus, in explaining the learning of programming, constructivist theory suits better than transmissive pedagogy because constructivist theory supports active learning while transmissive pedagogy supports passive learning.

Wulf and Hadjerrouit focused on constructivist approaches for teaching programming to overcome learning difficulties among novices [Wulf, 2005; Hadjerrouit, 2005]. In this research, we are interested in investigating the suitability of applying the constructivist theory of
Learning to the novice programmers’ learning tool because it supports active learning and the personalised learning style of different individuals.

3.5 Learning Styles

Novice styles of learning may contribute to success. Perkins et al. categorises learners as “stoppers” and “movers” [Perkins et al., 1989]. Learners in the stoppers category quit when they face problems or unclear direction. He suggested better instructional design to help learners, especially to aid stoppers once they are stuck on a problem. We would like to tackle the stoppers by having various different levels of programming questions, so that they can progress from an easy question to a more difficult question on the same topic at the same time. We choose to use multiple choice questions as a test instrument as learners may obtain answers and responses promptly. This should encourage learners to move on as they receive the answers of each question as soon as they try.

Related research has stated that the goal of teaching novices in basic programming courses is to embed deep learning into the learners [Robins et al., 2003]. Biggs’ Study Process Questionnaire (SPQ) [Biggs et al., 2001] seems to be a preferred test to study approaches to learning [D’Souza et al., 2008]. This questionnaire attempts to distinguish between the two main categories, namely deep and surface learners. Deep learners tend to maximise learning and surface learners usually rote learn and have a narrow target such as to pass the next examination.

There exist significant correlations between learning approaches and student performances, as emphasised by de Raadt et al. [de Raadt et al., 2005]. In addition, a study of 69 responses from computer science students has resulted in 22 who scored very highly by following a deep approach and 25 responses who scored very highly by following a surface approach [Jocelyn, 2005]. Therefore, based on Jocelyn’s work, there may exist two different high-scoring cohorts in a class: deep learners and surface learners. It is hard for the learning instructors to seek a balance in developing the course material to fit both groups.

In our own effort to understand the novice programmer, we conducted a survey based on the revised Two-Factor Study Process Questionnaire (R-SPQ-2F). We received and analysed 22 responses from the 268 students from Semester 1, 2008 enrolled in Programming 1.

The survey shows that 77% of the responses were from deep learners and 14% were surface learners (refer Table 3.6). Those who filled in the survey may be categorised as effective learners [Robins et al., 2003] because these groups of students seek the “extras” in
CHAPTER 3. COGNITIVE THEORIES OF LEARNING

Table 3.6: Responses to Biggs’ Revised Two-Factor Study Process Questionnaire (R-SPQ-2F)

<table>
<thead>
<tr>
<th>Learning Approaches</th>
<th>Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep</td>
<td>77</td>
</tr>
<tr>
<td>Balance</td>
<td>9</td>
</tr>
<tr>
<td>Surface</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
</tr>
</tbody>
</table>

order to succeed. Most of the effective students are in the deep learners category, however, because this is a small number, we are not able to assume that effective learners are deep learners.

Above all, we aim to minimise the gap in learning for these two groups, deep and surface learners, by providing different levels of programming questions with a range of difficulties. Deep learners may answer all the different levels of questions in order to learn, whereas surface learners may only answer minimal direct questions to construct their computer programming knowledge.

In the next section we outline the research groups that have contributed to the computer science education field. They have explored the learning theories in education in order to improve the teaching and learning of programming courses.

3.6 Theories of Learning for Programming Courses

Since 1986s, there are several studies related to learning theories in programming courses [Soloway, 1986; Olson et al., 1987]. These studies provide an understanding to scholars of how learning theories can be applied to teaching and learning computer programming. Improved teaching methods may lead to improved learning, enabling students to construct accurate mental models of programming concepts.

Researchers have grouped together to understand and alleviate difficulties of learning programming (see Section 2.2.1). We list some of the established groups that are actively doing research in this area in Table 3.7.

SIGCSE runs SIGCSE symposia, Innovation and Technology in Computer Science Education (ITICSE) and International Computing Education Research (ICER) conferences yearly to encourage researchers to share their new findings and attend workshops to discuss new research. Australasian Computing Education Conference (ACE) is held in cooperation of SIGCSE and ACM dedicated for Australasian region.

Other than those listed above, there are also small groups of researchers, such as the
McCracken group [McCracken et al., 2001] which studied novice competency and developed a trial assessment to investigate whether students can program. They presented evidence of students struggling to write programs due to a lack of knowledge and problem solving skills.

The Leeds group [Lister et al., 2004] discussed the novice students’ struggle to read programs due to their lack of problem solving skills. The Leeds group evolved into the BRACElet group [Lister et al., 2006]. The BRACElet group’s research interests include the analysis of students’ answers according to the SOLO Taxonomy and they plan to further investigate novice ability to read and write code [Clear et al., 2008].

3.7 Summary

This chapter highlights the taxonomies used in our research, setting the stage for their use in further work, to be discussed in subsequent chapters. We researched various educational theories that support learning to program. We discussed Bloom’s Taxonomy and explained the SOLO Taxonomy. We presented related studies in computer science courses that have applied these taxonomies.

We found that both of these taxonomies may be adapted to give useful insights into explaining why some novices struggle with particular programming tasks. These taxonomies may be used to classify questions based on complexity or difficulty to help novices to scaffold their cognitive processes. Furthermore, constructivist theory and learning styles may also
influence novice success in learning programming. We will use these educational theories in the development of our learning aid for novice programmers.

In Chapter 4 we explain our methodology used in undertaking this research. Then, in Chapter 5, we adapt the learning taxonomies discussed in this chapter to grade the levels of knowledge required to learn programming.
Chapter 4

Using Action Research to Understand Learning Difficulties

This chapter explains in detail our methodology for conducting this research. In Chapter 1 we highlighted that research in computer science education often involves surveys and testing to assist in developing learning platforms or tools. We surveyed novice programmers and instructors to obtain their insights about multiple choice questions in summative assessment. We conducted our research based on action research methodology. We applied it in five cycles in order to achieve our aim of probing novice minds through a proposed guided learning tool.

This chapter is structured as follows. In Section 4.1 we provide a general overview of the cycles in action research to be conducted during this research. In Section 4.2 we describe the theme in each cycle of the action research. In Section 4.3 we analyse the novice responses to multiple choice questions in the summative assessment. In Section 4.4 we look at questions and responses in a summative assessment plus instructor views about the complexity of multiple choice questions. In Section 4.5, we survey instructors to ask their views regarding assessment, multiple choice questions in assessment and their evaluation of multiple choice questions. In Section 4.6 and Section 4.7 we discuss and evaluate guided learning for novice programmers. In Section 4.8 we discuss the respondents’ profiles; including novices and programming instructors. In Section 4.10 we describe the summative assessments and in Section 4.11 we detail the instructor survey. Lastly, in Section 4.12 we describe the method of data analysis to be conducted in this research.
CHAPTER 4. USING ACTION RESEARCH TO UNDERSTAND LEARNING DIFFICULTIES

4.1 Action Research

Action research methodology has been employed in this thesis to take into account the variability of human responses. Action research supports researchers to improve their practice in their discipline [Morton-Cooper, 2000]. The improvements may include improving the practice; improving the understanding of a practice by practitioners and improving the situation in which the practice takes place [Tony, 2004]. Our research emphasise on the attempt to provide better understanding to teaching and learning in fundamental programming courses. Action research is a suitable methodology to provide iterative improvements in teaching and learning introductory programming [Thota et al., 2012].

Any action research study or project begins with one pattern of practices and understandings in one situation, and ends with another, in which some practices or elements of them are continuous through the improvement process while others are discontinuous (new elements have been added, old ones have been dropped, and transformations have occurred in still others). Similarly, understandings undergo a process of historical transformation. And the situation in which the practices are conducted will also have been transformed in some ways [Carr and Kemmis, 1986].

Action research is often used in the medical and educational areas of research [McIntosh, 2010] as both areas involve human responses and are constantly changing and improving their processes. Action research has been defined as a “practitioner-based” form of research in education as educators conduct the research in order to improve pedagogy and student learning [Phillips and Carr, 2010]. Practitioners apply action research in real situations, gaining feedback from experience, modifying the theory or model proposed as a result of this feedback, and trying it again [Muir, 2007]. It is also a critically-reflexive approach that generates a new way of “thinking, seeing and acting” [McIntosh, 2010] and a systematic approach to student learning in the classroom [Mills, 2010].

In this thesis, action research involves working with students and instructors in order to improve their skills, techniques and strategies in teaching and learning programming [Ferrance, 2000]. We depict one typical action research cycle in Figure 4.1, which involves five steps as listed below:

i. Defining the issue

ii. Planning the action
iii. Taking the action  
iv. Reflecting and refining  
v. Reporting the findings

Refering to Figure 4.1, the first step requires defining the issue. In this step, the researcher studies existing literature and makes general observations in the area of interest. Next, based on the observations and readings, the researcher carefully plans the approach to tackle the issues identified. In the third step the data collection will be conducted according to the plan. In the fourth step the data will be analysed and discussed. The fifth step is to write a report based on these findings. There is one step that bridges the first and second cycles. This step involves reflections of the analysed data from first cycle and feeding them as improvements to the research in to the second cycle. The researcher may improve their research by reflecting on the good points as well as “what went wrong” in the first cycle. These may contribute to better results and higher quality research in the second cycle.
CHAPTER 4. USING ACTION RESEARCH TO UNDERSTAND LEARNING DIFFICULTIES

The five steps in a cycle of the action research may be repeated in each cycle, depending on the researchers’ objectives. We have employed five cycles of action research for the research in this thesis. In the next section, we explain in detail the aims and each step taken in each cycle.

4.2 Applying Action Research Approach

In this section we outline the cycles involved and explain the methodology used to conduct this research (see Table 4.1). Each cycle uses a new group of respondents. For example, in Cycle 1, we analyse the questions and responses from a summative assessment sat by novices from semester two, 2007 and in Cycle 2, we analyse the questions and responses from a novice summative assessment from semester 1, 2008.

<table>
<thead>
<tr>
<th>Cycle Number</th>
<th>Theme</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice Programmers</td>
<td>To analyse questions and responses from novice summative assessment</td>
</tr>
<tr>
<td>2</td>
<td>Novice Programmers and Instructors</td>
<td>To analyse questions and responses from novice summative assessment plus instructor insights to the questions</td>
</tr>
<tr>
<td>3</td>
<td>Instructors</td>
<td>To survey instructors to find their views regarding summative assessment, multiple choice questions as the instrument in summative assessment and their evaluation of multiple choice questions.</td>
</tr>
<tr>
<td>4</td>
<td>Guided Learning 1</td>
<td>To introduce a formative assessment based on guided learning approach for novice programmers</td>
</tr>
<tr>
<td>5</td>
<td>Guided Learning 2</td>
<td>To improve the guided learning approach and introduce a guided learning tool</td>
</tr>
</tbody>
</table>

We conduct the research in five cycles. Each cycle is a progression from the previous cycle, except for the first. In the first cycle we begin with the programming novices as the theme because they are the centre of our research. We analyse their responses to the questions asked in a programming course’s summative assessment. We also then look at the questions, and in
particular, the ones where the novices struggled the most. In the second cycle we progress our research by analysing the questions and responses of a different summative assessment, but this time, we include the programming instructors’ views about classifying the questions and their difficulty levels. In the third cycle we survey all the programming instructors we can find to ascertain their opinions regarding summative assessments and their evaluations of the multiple choice questions. In the fourth cycle we introduce a guided learning approach that is aimed at assisting novices to learn programming and as a revision and preparation tool for the summative assessment. In the fifth cycle we improve the guided learning approach and tool based on the novices’ suggestions and we also add more questions to assess the more fundamental programming topics.

4.3 Action Research: Cycle 1

In the first cycle we focus on identifying the learning difficulties faced by novices through analysing their final examination sheets, including the instructor questions and novice responses. We outline the steps in this cycle in Table 4.2.

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Step Name</th>
<th>Descriptions</th>
<th>Refer</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Defining the Issue</td>
<td>Which programming concepts do novices find difficult to learn?</td>
<td>Section 2.2.1</td>
</tr>
<tr>
<td>ii</td>
<td>Planning Action</td>
<td>Analysing questions and answers in summative assessments to identify where novice learning difficulties occur.</td>
<td>Section 5.2</td>
</tr>
<tr>
<td>iii</td>
<td>Taking Action</td>
<td>Attempt to understand novice mental models of learning programming.</td>
<td>Section 5.2</td>
</tr>
<tr>
<td>iv</td>
<td>Reflecting and</td>
<td>Reflect on the data collected from the summative assessments.</td>
<td>Section 5.3</td>
</tr>
<tr>
<td></td>
<td>Refining</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>Report Findings</td>
<td>Detail findings.</td>
<td>Chapter 5</td>
</tr>
</tbody>
</table>

In the first step we define the focus as novices who face difficulties in learning to program. This issue has been discussed in Chapter 2. In Section 2.2 we highlighted the challenges
faced by novices in learning programming in terms of three entities of the Didactic Triangle: student, teacher and learning content.

Therefore, in the second step of this action research cycle, we aim to identify the particular learning difficulties through analysing the summative assessments. In the third step we analyse the questions in the assessment tasks (which we categorised according to Bloom’s Taxonomy). In the fourth step we reflect on the data collected from the summative assessments. We found it interesting to look at the level of complexity expected by the instructor and compare it to the responses given by the novices. In the fifth step we report our findings (Chapter 5).

In order to challenge the classifications based on the existing taxonomies, we study the questions tested in the summative assessment in Cycle 2 (Section 4.4).

### 4.4 Action Research: Cycle 2

In this cycle we analyse the responses to the multiple choice questions and a short answer question. The research approach is outlined in Table 4.3.

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Step Name</th>
<th>Descriptions</th>
<th>Refer</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Defining the Issue</td>
<td>Difficulty classifying questions and answers in programming assessments.</td>
<td>Section 4.3</td>
</tr>
<tr>
<td>ii</td>
<td>Planning Action</td>
<td>Study learning taxonomies in order to classify the questions and responses in summative assessments.</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>iii</td>
<td>Take Action</td>
<td>Analyse questions and responses in a summative assessment and instructor evaluation of the questions.</td>
<td>Section 5.4</td>
</tr>
<tr>
<td>iv</td>
<td>Reflecting and Refining</td>
<td>Reflect on the data collected from the summative assessment.</td>
<td>Section 5.5</td>
</tr>
<tr>
<td>v</td>
<td>Report Findings</td>
<td>Detail findings.</td>
<td>Chapter 5</td>
</tr>
</tbody>
</table>

The first step of the second cycle is to define the issue as being the problem of classifying
the levels of difficulty of questions and answers in a summative assessment. The details of the assessment can be referred to Section 4.10. In the second step we apply the existing learning taxonomies in order to classify the questions and responses in the summative assessment. Then, in the third step we analyse the questions and answers in the summative assessment based on the learning taxonomies. We also ask instructors to evaluate the questions in the summative assessment. The instructors are independent and do not know the novice responses. In the fifth step we elaborate on this classification as in Chapter 5.

As we reflect on the classification studied, we think that we need more information of instructor views in summative assessment because this cycle focuses on the novice perspectives. Therefore, we advance our research to survey the instructors in the next cycle.

4.5 Action Research: Cycle 3

Table 4.4: The Third Cycle

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Step Name</th>
<th>Descriptions</th>
<th>Refer</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Defining the Issue</td>
<td>Understanding the instructor perspectives towards assessment and multiple choice questions.</td>
<td>Section 4.4</td>
</tr>
<tr>
<td>ii</td>
<td>Planning Action</td>
<td>To devise a questionnaire for instructors.</td>
<td>Section 6.2</td>
</tr>
<tr>
<td>iii</td>
<td>Take Action</td>
<td>To survey instructors regarding summative assessment.</td>
<td>Section 6.4</td>
</tr>
<tr>
<td>iv</td>
<td>Reflecting and Refining</td>
<td>To reflect on the data collected from the instructors.</td>
<td>Section 6.6</td>
</tr>
<tr>
<td>v</td>
<td>Report Findings</td>
<td>Detail findings.</td>
<td>Chapter 6</td>
</tr>
</tbody>
</table>

In the third cycle, we first define the issue as being to understand the instructor views of summative assessment. In the second step we devise a questionnaire, discussed in detail in Section 6.2. In the third step we survey instructors to find their insights into summative assessments, in particular, how they consider multiple choice questions as a test instrument in programming assessments. We highlight the instructors’ expectations based on the difficulty
expected of the question. In the fourth step we discuss the data collected in Section 6.6. In the fifth step we detail the findings in Chapter 6.

Once the inputs from novices and instructors have been analysed, we apply our proposed classification and the relevant theories to develop a guided learning tool.

### 4.6 Action Research: Cycle 4

**Table 4.5: The Fourth Cycle**

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Step Name</th>
<th>Descriptions</th>
<th>Refer</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Defining the Issue</td>
<td>How to scaffold the learning of programming.</td>
<td>Section 7.1</td>
</tr>
<tr>
<td>ii</td>
<td>Planning Action</td>
<td>Help novices to learn programming through guided learning tool.</td>
<td>Section 7.2</td>
</tr>
<tr>
<td>iii</td>
<td>Take Action</td>
<td>To develop a guided learning tool for novices.</td>
<td>Chapter 7.2</td>
</tr>
<tr>
<td>iv</td>
<td>Reflecting and Refining</td>
<td>Survey and analyse the novices’ responses to the guided learning approach.</td>
<td>Chapter 8.2</td>
</tr>
<tr>
<td>v</td>
<td>Report Findings</td>
<td>Detail findings.</td>
<td>Chapter 7 and Chapter 8</td>
</tr>
</tbody>
</table>

In the fourth cycle we first aim to develop a guided learning tool based on the analysis of assessments and instructor perspectives. In the second step, we use the learning taxonomies to classify the questions in order to scaffold the learning of programming and build a guided learning approach. Next we develop a guided learning tool which consists of a few classifications of programming tasks. In the fourth step we ask novices to evaluate the learning tool. In the fifth step we detail and reflect on our findings in Chapter 7 and 8.

### 4.7 Action Research: Cycle 5

The fifth cycle aims to build on the fourth cycle of our research, but with a new group of novice programmers. At the end of the fourth cycle we collect the novice insights into the
CHAPTER 4. USING ACTION RESEARCH TO UNDERSTAND LEARNING DIFFICULTIES

guided learning tool. In this cycle we aim to improve the guided learning tool based on the novices’ recommendations from cycle 4. We then report the novice responses and use them to improve the guided learning tool. Next, we evaluate novice responses to the guided learning tool. Last but not least, we report our findings in Chapter 7 and our conclusions and reflections in Chapter 8.

Table 4.6: The Fifth Cycle

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Step Name</th>
<th>Descriptions</th>
<th>Refer</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Defining the Issue</td>
<td>Improving guided learning tool.</td>
<td>Section 7.3</td>
</tr>
<tr>
<td>ii</td>
<td>Planning Action</td>
<td>Analyse novice responses to improve guided learning tool.</td>
<td>Section 7.3</td>
</tr>
<tr>
<td>iii</td>
<td>Take Action</td>
<td>To improve the guided learning approach based on novices’ recommendations</td>
<td>Section 8.5</td>
</tr>
<tr>
<td>iv</td>
<td>Reflecting and Refining</td>
<td>Evaluate the novices’ responses to the guided learning tool.</td>
<td>Section 8.5</td>
</tr>
<tr>
<td>v</td>
<td>Report Findings</td>
<td>Detail findings</td>
<td>Chapter 7 and Chapter 8</td>
</tr>
</tbody>
</table>

4.8 Respondent Profiles

We divide our respondents into two main categories: novices and instructors. In this section, we provide definitions of the novices and instructors.

4.8.1 Novice

The term *novice* in this research is defined as any individual who is formally enrolled in university classes to learn the fundamentals of programming concepts with little or no programming background. We focus on tertiary education in the School of Computer Science, RMIT University because foundation programming courses are the pillar of their learning and the students need to have strong basics in programming concepts in order to progress successfully from one semester to the next. In some contexts, we may also use the term
“student” or “learner” to be referred as a novice.

They may be from Programming 1 (P1), which is a core, first-year course in every undergraduate degree, hence providing a high volume of representative data. This course is aimed to teach students to code simple, small Java programs involving, at their most challenging stage, a moderate-sized system with a range of classes (and simple inheritance requirements).

“Programming 1 course contributes to the development of the following graduate capabilities, including:

Enabling Knowledge: Syntax and basic features of the object-oriented programming language Java; good programming style, standards and practices in programming; the use of standard Java classes, interfaces, containers; and basic techniques for code reuse and testing” [D’Souza, 2006].

Programming 1 course is aimed to expose novices to the knowledge of syntax and fundamentals of Java language, we are hoping to aid novices to achieve this goal by proposing a learning tool for the first four weeks of the learning contents or modules in a semester.

Novices may be advised to complete a prerequisite course (Introduction to Programming) before they can enrol to Programming 1 course, if they have weak logic skills and below expected marks in the mathematics pre-requisite courses. They are assessed based on their performances in high schools or during the previous semester. The Introduction to Programming course emphasises logic and pseudocode and mostly uses Alice (available at: http://www.alice.org/) as a platform to deliver the course content. Table 4.7 describes the Introduction to Programming and Programming 1 courses’ contents in detail.

4.8.2 Instructor

The term instructor in this research is defined as any individual who is lecturing, teaching or tutoring programming courses. They must have experience in foundation level programming courses in tertiary education, and is irrespective of whether or not they are a lecturer, professor or tutor. In the survey in Chapter 6 we also include instructors from secondary levels of education, as long as they are involved in teaching programming courses. Furthermore, we include in this definition anyone involved in the preparation of test questions, final examinations or practical work assessment instruments.

In the next section we explain the details of the programming courses involved in this research.
4.9 Details of Programming Courses

We gather data from both novices and instructors of programming courses. These courses may include Introduction to Programming (IP), Programming 1 (P1), Programming 2 (P2), Programming 3 (P3) and Java for Programmers (JP). Brief descriptions of these courses appear in Table 4.7, along with their prerequisite courses. We do not take into account the choice of programming language for the programming courses.

Table 4.7: Programming Courses

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Descriptions</th>
<th>Pre-requisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to Programming</td>
<td>This course covers the foundations of computer systems and programming, which are the conceptual building blocks for programming, usually caters for non-Computer Science or IT related majors.</td>
<td>Zero</td>
</tr>
<tr>
<td>Programming 1 (P1)</td>
<td>This course covers introductory programming concepts, basic structures and algorithms for novices.</td>
<td>IP or Zero</td>
</tr>
<tr>
<td>Programming 2 (P2)</td>
<td>This course extends the learning of programming to cover more difficult programming principles.</td>
<td>P1</td>
</tr>
<tr>
<td>Programming 3 (P3)</td>
<td>This course covers a detailed study of programming; including the use of defensive programming, debugging, testing, coding standards and practices.</td>
<td>P1 and P2</td>
</tr>
<tr>
<td>Java for Programmers (JP)</td>
<td>This course contributes to the development of the following capabilities: enabling knowledge, problem solving, and critical analysis.</td>
<td>IP</td>
</tr>
</tbody>
</table>

4.10 Descriptions of the Summative Assessment

In this section we describe in general the distributions of marks for the summative assessment of programming courses in RMIT University. The contribution of the summative assessment to the overall final grade is 50%. The number of total marks for this summative assessment is 100. Novices are required to answer all of the questions in the space provided in the examination booklet.
In summative assessment, it is assumed that all basic programming concepts have been taught, however different institutions may arrange their course documents differently, and so the content may be covered in lectures, in tutorials, in laboratories, or indeed in all of these, and in different contexts and in different chronological order.

The structure of the summative assessment is as follows:

- Part 1 is worth a total of 30 (out of 100) marks and contains 20 equally weighted short answer questions but the majority are multiple choice questions. For each multiple choice question there is only one correct answer and three incorrect answers (distractors).
- Part 2 is worth 35 marks and contains seven equally weighted questions requiring code writing for short programs or parts of programs.
- Part 3 is worth 35 marks and requires incremental development of code for a single problem context, which tests novices’ applications of simple object-oriented code completion or development from scratch. We provide no further details as this part of the examination was beyond the scope of the research reported here.

The time allocated to complete the paper was three hours plus an extra 15 minutes of reading time. On this basis, the expected estimated time to answer each multiple choice question was approximately two to three minutes. There were no “explain in plain English” type of questions. However, sections of Part 2 required the recognition of code segments to be altered or extended.

The exam paper was prepared by an experienced team of instructors in the second half of the semester, during which time the novices were enrolled in the course. This allowed the instructors to adapt the paper to the current context of course delivery, as needed. During the semester a range of preparatory instruments and activities were used to encourage novices to become more conversant with the style and complexity of the final examination paper. These activities included the following: tutorial and laboratory exercises, a mid-semester test in Week 8, to test material covered up to and including Week 6 (the halfway point the 12 week semester); an ongoing series of Weblearn quizzes and tests; and discussions in lectures of model questions and answers from past exam papers, comparable to the style and complexity of what might be expected in their final exam. Weblearn is a school developed online system for the management of question banks for the courses taught.
CHAPTER 4. USING ACTION RESEARCH TO UNDERSTAND LEARNING DIFFICULTIES

4.11 Descriptions of Instructor Survey

We design a questionnaire to gather data about instructors’ general views of summative assessments and, more specifically, to gauge their views about the appropriateness of multiple choice questions as assessment instruments for summative assessment.

We approach programming instructors all over the world through special interest group emailing lists and staff email lists. The lists include the Special Interest Group of Computer Science Education (SIGCSE) which includes the Innovation and Technology in Computer Science Education (ITiCSE) group and the Psychology of Programming Interest Group (PPIG) (shown in Table 3.7). Participation in the questionnaire is voluntary and anonymous, therefore, we are unable to determine the location of the respondents, we only know the research groups that they joined, but some may have joined more than one. As they are all considered instructors, we do not find it crucial to analyse the data collected based on the research groups, but rather simply as a group of instructors.

The target list of potential respondents was intentionally chosen this way to maximise input from informed educators in the context of teaching novice programming. Moreover, we decided to implement the survey as an online form so as to assure respondents that they would indeed remain completely anonymous. We believe these choices will ensure reliable feedback from the respondents. Finally, the various groups are well known and provide us with a potentially diverse set of respondents. SIGCSE has approximately 1000 members subscribed to the mailing list. PPIG alone, for example, is the smallest of the three groups in our target list, yet they report via their website an approximate membership of 300 worldwide, which brings together people from diverse communities who share a common interest in the psychological aspects of programming. The other two groups are more widely known in the computer science education context and have well established, global memberships.

4.12 Method of Data Analysis

"With statistics, we can summarize large bodies of data, make predictions about the future trends and determine when different experimental treatments have led to significantly different outcomes. Thus, statistics are among the most powerful tools in the researcher’s toolbox" [Leedy and Ormrod, 2004].

In this research, data were analysed using a statistical software package called Statistical Package for Social Science (SPSS) Package 15.0. SPSS is a comprehensive system for
CHAPTER 4. USING ACTION RESEARCH TO UNDERSTAND LEARNING DIFFICULTIES

analysing data [Levesque, 1998; Norusis, 2006; Zagumny, 2001]. SPSS can take data from almost any type of file and use them to generate tabulated reports, charts and plots of distributions and trends, descriptive statistics and complex statistical analyses.

The descriptive statistics for ranking of variables used in this research are:

- The frequency of data collected to analyse detailed information on nominal (category) data and to describe the results.
- Mean of data a method of calculating the centre of the data set [Carroll, 2007]. Mean represents a whole data set of scores with a single number.
- Index of discrimination is a measure of the differences in scoring on various questions between students who have performed well and poorly [Levesque, 1998].
- Item analysis is in depth analysis of the questions based on the responses to multiple choice questions.
- Crosstabulation is a joint frequency distribution of cases based on two or more categorical variables [Michael, 2002]. It is used to show the distributions of responses between the crosstabulated variables. It can be a combination of two or more frequency of data, usually presented in table form.
- Correlation Classification
  The explanations for correlation classifications will be elaborated in the next subsection.

4.12.1 Correlation Classification

The SPSS software package is able to automatically determine if there exist any correlations between the variables listed based on the observed significance level of the test that we set.

“Correlation is the statistical concept which describes the amount and type of relationship between two variables. Correlations describe the relation between two variables and how that relationship functions, whether it is a positive relationship (direct) or negative relationship (inverse)” [Malloy, 2000].

Correlation is a statistical process by which the researcher discovers the nature of relationships among different variables. There are two important components which need to be
clarified to determine the relationships between the variables tested; the coefficient of correlation and the significance of the correlation. We will discuss these components in the next subsections.

**Coefficient of Correlation**

A correlation coefficient indexes two properties of a relationship [Williams and Monge, 2001]. The two properties are:

- **Strength**
  The degree to which the variables vary together. It is indicated by the size of the correlation coefficient. A correlation of +1 or -1 indicates a perfect correlation while a correlation of 0 indicates no correlation.

- **Direction**
  The direction of the relationship is indicated by the sign of the correlation coefficient, in other words, by whether the number is a positive or negative one. If two variables tend to move up or down together, they are said to be positively correlated and if they tend to move in opposite directions, they are said to be negatively correlated.

The correlation coefficient statistical analysis in our research is based on the information in Table 4.8. For example, if a correlation of +0.21 is found, this means there is a weak, positive relationship between the variables tested.

**Table 4.8: The Strength and Direction of Correlation Coefficient Classifications by Rowntree [Rowntree, 1981]**

<table>
<thead>
<tr>
<th>Negative Range</th>
<th>Description</th>
<th>Positive Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>No correlation</td>
<td>0.00</td>
</tr>
<tr>
<td>(-0.01) to (-0.19)</td>
<td>Very Weak</td>
<td>0.01 to 0.19</td>
</tr>
<tr>
<td>(-0.20) to (-0.39)</td>
<td>Weak</td>
<td>0.20 to 0.39</td>
</tr>
<tr>
<td>(-0.40) to (-0.69)</td>
<td>Moderate</td>
<td>0.40 to 0.69</td>
</tr>
<tr>
<td>(-0.70) to (-0.89)</td>
<td>Strong</td>
<td>0.70 to 0.89</td>
</tr>
<tr>
<td>(-0.90) to (-0.99)</td>
<td>Very Strong</td>
<td>0.90 to 0.99</td>
</tr>
<tr>
<td>-1.00</td>
<td>Perfect</td>
<td>1.00</td>
</tr>
</tbody>
</table>

**Significance of Correlation**

In addition to the magnitude and direction of a correlation coefficient, there are also methods to test the significance of the correlation. Hypothesis testing is tested on two conditions; the
null hypothesis and the research hypothesis or alternative hypothesis.

In order to do the hypothesis testing, the population product-moment correlation coefficient is used to observe the significance level of the variables tested. The population product-moment correlation coefficient is usually denoted by the Greek letter $\rho$ [Upton and Cook, 2000].

These two conditions (research hypothesis and null hypothesis) are decided on the condition that the $\alpha$ level is set to be zero. Therefore, if $\rho$ is equal to zero, then the variables are said to be uncorrelated.

Hypothesis testing is done under the assumption that the null hypothesis is true, meaning that no change happened. If $\rho$ is not equal to zero, then the null hypothesis is rejected and therefore research hypothesis is accepted.

The null hypothesis is written as; $H_0: \rho = 0$.

The research hypothesis is written as; $H_1: \rho \neq 0$.

The probability value (also called the p-value) is the probability that the result found in the research study will occur (or an even more extreme result will occur), under the assumption that the null hypothesis is true.

The significance level ($\alpha$ level) is set based on the confidence level of the tested variables and is applied to decide when to reject the null hypothesis. Most psychologists and scientists set the $\alpha$ level at 0.05 [Hoffman, 2002]. Hence, this will reject the null hypothesis when the p-value is less than or equal to 0.05. This is to allow a small number of errors to occur [Fisher, 1970; Dallal, 2008]. In this research, we also set the $\alpha$ level to 0.05. Therefore, if p-value $\leq$ 0.05, there exists a significant correlation.

4.13 Summary

As explained in Section 4.1, we decided to use action research methodology because it is a well established form of research into learning and technology. We plan to use five cycles of data gathering to establish the research findings.

In the first cycle we classify the questions and responses in two sets of novice summative assessments. In the second cycle we survey the questions and novice responses plus the instructor views of the questions in the summative assessment. In the third cycle we look at the instructor perspectives of the questions tested in summative assessments. We would like to understand the instructor expectations to novices of the questions tested in the assessment. The data gathered from these three cycles will benefit us in developing a learning tool, which
will be discussed in detail in Chapter 7.

We survey novices regarding the guided learning tool in two cycles (cycle four and five) to validate our research outcome. In the subsequent chapters we will explain in detail the results and discussions regarding novices, instructors and the proposed learning tool.
Chapter 5

Novice Responses to Summative Assessment

In the previous chapter we have explained that we will conduct five cycles of action research in order to establish the development of a learning tool for novice programmers. This chapter presents the data collected from the first and second cycles of action research. We study the multiple choice questions and a short answer question from two sets of Programming 1 course’s summative assessments. In the first cycle, we analyse novices’ overall and individual responses to multiple choice questions, and explore the classification of the multiple choice questions and novices’ responses in terms of Bloom’s Taxonomy. In the second cycle, we classify the multiple choice questions based on two taxonomies of learning; Bloom’s and SOLO taxonomies (explained in Chapter 3), and conduct index of discrimination and item analysis (as explained in Chapter 4). We also propose classifications for multiple choice questions to support the inadequacy in existing learning taxonomies.

This chapter is structured as follows. Section 5.1 highlights our motivation to study summative assessments and in particular the usage of multiple choice questions in summative assessment for novices. Section 5.2 covers our analysis of the first set of summative assessment. We report overall and individual responses to each multiple choice question. Then, we classify the multiple choice questions based on our modified version of Bloom’s Taxonomy to cater for the needs of Computer Science courses and discuss possible responses to the multiple choice questions. Section 5.3 provides our reflections on the data gathered to direct us for further research in the second cycle of action research.

The following sections are derived from the second cycle of action research. Section 5.4
details our analysis of novices’ assessments. In this cycle, we analyse the responses from a different group of students. We classify the multiple choice questions based on Bloom’s Taxonomy and our newly defined classification of instructor levels of complexity and novice levels of difficulty. We calculate the index of discrimination and conduct item analysis on novice responses in summative assessment. Then we explain the significance between the four classifications discussed in this section (Bloom’s Taxonomy, instructor levels of complexity, novice levels of difficulty and index of discrimination). In addition, we categorise a code writing question based on the SOLO Taxonomy. Section 5.5 discusses the findings from the data collected.

5.1 Motivation: Analysis of Multiple Choice Question in Summative Assessment

“I know the material, but when I take the test I go blank!”

[Danskin and Lambert, 1989]

In Section 2.2 we highlighted that novices face challenges to learn programming. In academia, assessment plays the role of the key indicator for students’ performances in the course [Garfield, 1994]. Referring to Figure 2.1 novices may not perform well in their fundamental programming assessments. In this chapter we would like to explore the summative assessment or final examination questions and responses to understand instructors’ expectations for each multiple choice question and the learning difficulties novices may show through choosing the wrong answer.

We attempt to classify the questions and novice responses in an introductory programming course. Our focus is on the multiple choice question section of the summative assessment for the Programming 1 course. Our main goal is to elucidate the idea of novices’ mental models of learning fundamental concepts of programming and to understand instructors’ expectations for each multiple choice questions tested in the summative assessment.

In order to achieve our goal, we report the analysis of the questions and responses for the Programming 1 summative assessment. We also classify novices’ responses to a short answer question. We analyse the questions and novices’ responses in two cycles of action research methodology.

In the first cycle, we analyse overall and individual novice responses to the multiple choice questions. Then, we classify and discuss three multiple choice questions and the four possible
responses for each question in order to learn how novices attempt to choose the answer for the questions. We discuss the application based on Bloom’s Taxonomy.

In the second cycle, we not only classify the multiple choice questions based on Bloom’s Taxonomy, but we also identify the instructor levels of complexity, novice levels of difficulty, conduct index of discrimination and item analysis.

The questions discussed in this chapter are attached in Appendix C. The questions we discuss are chosen because they best represent a set of questions that are capable of evaluating novice programmers’ skills.

Our research is different from previous studies that analyse novice programmers’ responses [Lister et al., 2006; Whalley et al., 2006; Thompson, 2007; Lopez et al., 2008] in the following ways:

- We look at real programming tasks drawn from the summative assessment sat by novice programmers and classify them according to Bloom’s and SOLO taxonomies.
- We look at both multiple choice questions and a short answer coded question.
- We classify the questions based on Bloom’s Taxonomy, and identify the instructor levels of complexity, novice levels of difficulty, and conduct index of discrimination and item analysis to the multiple choice questions in the summative assessment.

Overall, we aim to explore the questions and responses tested in novices’ summative assessment to build our understanding of novices’ mental models of fundamental programming.

5.2 A Study of Multiple Choice Questions in Summative Assessment

The summative assessment of the Programming 1 course tests fundamental concepts of programming. In this section, we explore the questions and novice responses in the summative assessment and from these, extract or build up our understanding of some of their mental models of programming. We focus mainly on understanding novice responses to multiple choice questions and discussing the models presented in the alternative answers (distractors).

In the first cycle of action research, we review 89 exam answer sheets from the summative assessment of semester 2, 2007. We include the questions in Appendix C.1.1. The solutions presented in examinations are devoid of any collaborative context, so errors in examinations are far more likely to be revealing of programming difficulties. There are 20 questions in Part 1 of the assessment sheets. The details of the assessment are in Section 4.10. Although there
are 20 questions, we selected 16 of them to review. We exclude four questions because one is a “fill in the blanks” question and is inappropriate for this part of the study where we are focusing purely on multiple choice questions. The three remaining questions were excluded because there were doubtful answers in the options and free marks had been awarded to the students.

5.2.1 Overall Responses to Multiple Choice Question

Table 5.1: Overall Responses to Multiple Choice Questions

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Option (in %)</th>
<th>Total (in %)</th>
<th>Correct Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.37</td>
<td>59.55</td>
<td>34.83</td>
</tr>
<tr>
<td>2</td>
<td>7.87</td>
<td>22.47</td>
<td>61.80</td>
</tr>
<tr>
<td>3</td>
<td>7.87</td>
<td>61.80</td>
<td>20.22</td>
</tr>
<tr>
<td>4</td>
<td>12.36</td>
<td>16.85</td>
<td>61.80</td>
</tr>
<tr>
<td>5</td>
<td>12.36</td>
<td>22.47</td>
<td>53.93</td>
</tr>
<tr>
<td>6</td>
<td>25.84</td>
<td>64.04</td>
<td>6.74</td>
</tr>
<tr>
<td>7</td>
<td>10.11</td>
<td>20.22</td>
<td>4.49</td>
</tr>
<tr>
<td>8</td>
<td>34.83</td>
<td>51.69</td>
<td>2.25</td>
</tr>
<tr>
<td>9</td>
<td>48.31</td>
<td>28.09</td>
<td>15.73</td>
</tr>
<tr>
<td>11</td>
<td>15.91</td>
<td>17.05</td>
<td>46.59</td>
</tr>
<tr>
<td>12</td>
<td>34.83</td>
<td>59.55</td>
<td>2.25</td>
</tr>
<tr>
<td>13</td>
<td>10.11</td>
<td>42.70</td>
<td>12.36</td>
</tr>
<tr>
<td>14</td>
<td>16.85</td>
<td>26.97</td>
<td>37.08</td>
</tr>
<tr>
<td>15</td>
<td>1.12</td>
<td>4.49</td>
<td>0.00</td>
</tr>
<tr>
<td>16</td>
<td>4.49</td>
<td>10.11</td>
<td>17.98</td>
</tr>
<tr>
<td>17</td>
<td>44.32</td>
<td>2.27</td>
<td>4.55</td>
</tr>
</tbody>
</table>

First, we outline the overall performance of novices in the summative assessment. Table 5.1 reports the novice responses for each multiple choice question and the correct answer for the questions. The correct response for each question is highlighted in yellow.

Question 17 can be labelled as the easiest question because 94.83% of the novices chose the correct answer. Questions 14 and 15 have the least correct responses from novices, 34.83%. For Question 14, almost 60% of the responses chose distractor B. Question 16 has 37.08% of correct responses. Other questions have between 40% to 70% of correct responses. Table 5.1 accumulates the overall responses, and in the next subsection we analyse the individual responses in an attempt to measure novice abilities to program.
5.2.2 Individual Responses to Multiple Choice Questions

We conduct item analysis [Baker and Kim, 2004] to report individual responses to multiple choice questions. In this cycle of research, the item analysis measures the students’ proficiency based on the number of items (or referred to as questions) tested in an assessment. Item analyses are frequently conducted on multiple choice questions as the test instrument [Yu, 2010] because multiple choice questions are usually scored dichotomously; the correct answer receives a score of one and each distractor (the wrong answer) yields a score of zero. Therefore, it can be easy to analyse them based on “right” or “wrong” answers only.

In Table C.1, Table C.2 and Table C.3 (refer to Appendix C.2), we outline the 89 novices’ responses to the multiple choice questions in a set of final examination answers for the Programming 1 course. \(s1\) represents student number 1, \(s2\) represents student number 2 and the same apply to the remaining. \(q1\) represents question number 1, \(q2\) represents question number 2 and so on. We outline the individual responses for each question, total score and average score (total score divide by number of question) for each novice’s responses to the multiple questions, horizontally. At the end of Table C.3, we provide the total score for each question and the average score for the particular question. Each multiple choice question is awarded one mark.

As each question is equally weighted (one mark for each correct response, and no mark awarded for wrong answer), we calculate the average score and this score will represent the novice’s proficiency in fundamental programming concepts based on the sixteen multiple choice questions tested in the summative assessment. Only one novice, \(s28\), manages to answer all 16 questions correctly, and is therefore considered as possessing 100% proficiency with the fundamental programming concepts. Novices \(s18\), \(s27\), \(s39\), \(s49\), \(s67\) and \(s76\) possess 88% proficiency with the fundamental programming concepts required by the programming instructors who devised the questions.

In Table 5.2, we group the novices who scored 88% proficiency with the fundamental programming concepts. Although they scored the same average they responded to the multiple choice questions differently, for example, \(s27\) answered 14 questions correctly and but did not choose the correct answer for \(q11\) and \(q14\), while \(s39\) answered 14 questions correctly but did not choose the correct answer for \(q1\) and \(q16\). All the six novices who scored 88% proficiency with the fundamentals of programming chose the wrong answer for different questions, but four of them did not get the correct answer to Question 14. Referring to Table 5.1, Question 14 has one of the least numbers of correct responses in this set of questions.
In another group of the sample, there were eight novices who scored 0.31 for the set of multiple choice questions (refer to Table 5.3). Although they scored the same, 31% of proficiency, we cannot draw a firm conclusion that they have the same level of proficiency because \(s38\) answered \(q1, q2, q4, q17\) and \(q18\) correctly and the others responded differently to the 16 questions.

None found the correct answer for \(q9\) and \(q15\). Four novices managed to choose the correct answer for \(q14\). Question 14 and 15 have the least number of correct responses for this set of multiple choice question.

Hence, we cannot judge a student’s abilities in programming courses based on the number of correct responses in the assessment. Rather, the classifications of the observable knowledge and abilities for each question should also be taken into account. In the next section, we attempt to classify multiple choice questions and novices’ responses based on our existing learning taxonomy.
5.2.3 Analysing Multiple Choice Questions Based on the Modified Version of Bloom’s Taxonomy

The objective of the item analysis is to analyse the students’ responses to the multiple choice questions. Novices may have the same score, but may respond differently to the same set of multiple choice questions. We discover that there is an inadequacy in describing the skills tested in the assessment using item analysis.

Based on our literature survey as discussed in Chapter 3, Bloom’s Taxonomy was developed for describing the level of cognitive process involved in learning. However, there can be many categories in a learning taxonomy, as learning and background information are wide and varied. We find that it is difficult to categorise the cognitive levels based on the correct answer, let alone the other multiple options provided in the question. The Bloom’s Taxonomy categories were not developed to relate to programming questions, and knowing which “keyword” in each category to apply to the question at hand is not clear. Hence, we reconsider each category in the revised Bloom’s Taxonomy to suit the nature of programming course (refer to Table 5.4).

Table 5.4: Learning Taxonomies based on Revised Bloom’s Taxonomy [Anderson et al., 2001] and our own Modified Version of Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Level</th>
<th>Revised Bloom’s Taxonomy</th>
<th>Modified of Bloom’s Taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Create</td>
<td>Create</td>
</tr>
<tr>
<td>5</td>
<td>Evaluate</td>
<td>Analyse</td>
</tr>
<tr>
<td>4</td>
<td>Analyse</td>
<td>Apply Implement</td>
</tr>
<tr>
<td>3</td>
<td>Apply</td>
<td>Apply Execute</td>
</tr>
<tr>
<td>2</td>
<td>Understand</td>
<td>Understand</td>
</tr>
<tr>
<td>1</td>
<td>Remember</td>
<td>Remember</td>
</tr>
</tbody>
</table>

We present our modified version of Bloom’s Taxonomy in Table 5.4, where column 3 is based on the revised version of Bloom’s Taxonomy (Table 5.4, column 2). We aim to apply the revised Bloom’s Taxonomy to the examination questions and novice responses. We maintain levels 1 and 2 as remember and understand the programming concepts, but we expand the apply level to two separate levels: level 3 - apply execute and level 4 - apply implement. Level 5 will be analyse to replace the original evaluate level of the revised Bloom’s Taxonomy and level 6 is the same, create.

Johnson and Fuller stated that application level of Bloom’s Taxonomy is the aim of computer science teaching [Johnson and Fuller, 2006]. Therefore we believe this modification
is more appropriate for programming courses as we want to emphasise the application level of Bloom’s Taxonomy. In particular, we perceive the difference between the categories of implement and execute within the apply category. We consider that to implement a task involves a higher level of understanding than to simply trace through some lines of code and execute them as a computer would. Also, within the context of programming, the category of analyse (level 5) is often taken to mean, “find the errors” which can be very difficult and so we have put this at the next level up in the cognitive hierarchy. Finally, level 6, create applies to actually writing program code from scratch. Usually the highest categories, level 6 in this instance, are not tested by multiple choice questions; create can be tested and seen in the programming project, whereby students are able to practise generating, planning and production skills throughout the semester.

Next we describe three multiple choice questions (questions 5, 11 and 14) and the responses from the summative assessment according to the Bloom’s Taxonomy. We choose these three questions because they best explain our application of different levels of Bloom’s Taxonomy to the questions and responses. We analyse novice mental models in selecting the options provided to each multiple choice question. We attempt to explain the mental model that novices may achieve if they manage to find the correct answer and attempt to explain their thinking if they choose the distractors.

**Question 5**

Question 5 tests on shorthand operators. Novices learn about this topic in the second week of the Programming 1 course.

\[
y = x + x;
\]
\[
y += y + y;
\]

are equivalent to:

A. \( y = 2 \times x \)
B. \( y = 4 \times x \)
C. \( y = 6 \times x \)  \(\text{(Answer)}\)
D. \( y = 8 \times x \)

We classify the question as level 3 (apply execute) in our modified version of Bloom’s Taxonomy. Therefore, if the novice is able to correctly answer this question, the novice achieves the apply execute level being tested by such a question.
CHAPTER 5. NOVICE RESPONSES TO SUMMATIVE ASSESSMENT

Question 11

Question 11 covers the concept of passing parameters covered in week six of the semester. We suggest that the level of difficulty for this question is *apply implement* in our modified Bloom’s Taxonomy and set it at level 4.

What are the values of the variable a, b and c after execution of the following program:

```java
public class TestParameterPassing {
    public static void main (String [ ] args) {
        int a = 20;
        double b = 1.5;
        int c;
        c = aMethod (b,a);
    }

    public static int anyMethod (double x, int y) {
        x = x * 2;
        y = y + 10;
        return (int) (x * y);
    }
}
```

A. a = 20; b = 1.5; c = 90 (Answer)
B. a = 30; b = 3.0; c = 90
C. a = 20; b = 1.5; c = 30
D. a = 30; b = 3.0; c = 30

Question 14

In question 14 novices are being tested on the concepts of a multi-dimensional array covered in week nine of their 13 week semester. This question is one of the questions that has the least number of novices choosing the correct answer. Hence, it is important to study the possible misconceptions that may occur.

What is the output of the program below?

```java
public class void main (String [ ] args) {
    int [ ] [ ] m = new int [3] [3];
}
```
for (int row = 0; row <= 2; row++) {
    for (int col = 0; col <= 2; col++) {
        m[col][row] = row;
    }
    for (int i = 0; i<3; i++) {
        for (int j = 0; j<3; j++) {
            System.out.print(" "+m[i][j]);
            System.out.println();
        }
    }
}

A. 0 1 2 (Answer)
   0 1 2
   0 1 2

B. 0 0 0
   1 1 1
   2 2 2

C. 1 1 1
   2 2 2
   3 3 3

D. 1 0 0
   0 1 0
   0 0 1

We suggest that the level of difficulty for this question is analyse in Bloom’s Taxonomy and so have set the correct answer A as level 5 in our modified classification of Bloom’s Taxonomy.

5.2.4 Novice Responses to Multiple Choice Questions

In the previous subsection, we discussed the modified version of Bloom’s Taxonomy and provided a few examples of our application of the learning taxonomy. In this section, we attempt to elucidate the idea of novice mental model in choosing that answer from the list of options provided by the instructors.

Novices are advised to evaluate the possible answers (A, B, C or D) when answering multiple choice questions. If the novices selected the wrong option, we can say their cognitive processes are found to be lower than expected by the examiners (instructors). We present a summary of our analyses of the three multiple choice questions in Table 5.5, Table 5.6 and
As this question was covered in the early weeks of the semester, we would expect a high percentage of correct responses, but only 53.93% chose the correct answer (option C). This shows that little more than half of the responses were able to meet the instructor expectations of their understanding of variables and operations, a minimal requirement for novice programmers. There were 12.36% responses which chose option A, 22.47% responses which chose option B and 11.24% chose option D.

This question is direct and it requires the student to understand variables and operations. We have classified it at the third level of Bloom’s Taxonomy, *apply execute*. Option C is the correct answer for this question. This option shows that the novice is able to apply the shorthand operator. Thus we rate them at level 3 as expected by the instructors.

Novices who chose options A or B, may have misunderstood the concept of a shorthand operator. If the novices have grasped the concept of variable storage, we would assume...
they should understand the first statement \( y = x + x \); would be equivalent to \( y = 2 \times x \). However, if the novices have no concept of what the \( + = \) operator is performing, they might be tempted to settle for what they consider to be halfway to the correct answer, say option A. Novices who chose this option, have the understand level of Bloom’s Taxonomy, a minimal requirement of the cognitive levels. The next two options are at higher levels, since the novices may have tried to consider what the \( + = \) operator is performing. Both options B and D indicate a misinterpretation of how the operator works, but option D is an answer which amounts to little more than a guess. Hence we have classified the levels of the cognitive processes involved for options A and B as not able to achieve level 2 (understand), while answer D is unable to achieve level 1 (remember) and could have been a slip in reading and executing the equation.

**Question 11**

Table 5.6: Analysis of the Novice Responses for Question 11

<table>
<thead>
<tr>
<th>Option</th>
<th>Possible Feedback</th>
<th>Descriptions</th>
<th>Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a = 20; b = 1.5; c = 90</td>
<td>Able to distinguish the difference between ( a ) and ( x ), ( b ) and ( y ) and able to return the correct value of ( c ) by calling a method</td>
<td>48.31%</td>
</tr>
<tr>
<td>B</td>
<td>a = 30; b = 3.0; c = 90</td>
<td>Mistakenly give the value of ( x ) and ( y ) as ( a ) and ( b ) and able to return the value of ((\text{int}) (x \times y))</td>
<td>28.09%</td>
</tr>
<tr>
<td>C</td>
<td>a = 20; b = 1.5; c = 30</td>
<td>The value of ( b ) is correct, however, novices return the value as ((\text{int}) (a \times b))</td>
<td>15.73%</td>
</tr>
<tr>
<td>D</td>
<td>a = 30; b = 3.0; c = 30</td>
<td>Mistakenly give the value of ( x ) and ( y ) as ( a ) and ( b ) and return the value of ((\text{int}) (a \times b))</td>
<td>7.87%</td>
</tr>
</tbody>
</table>

Only 48.31% of the responses chose the correct answer for Question 11 (option A), 28.09% of the responses chose option B, 15.73% chose option C and 7.87% chose option D.

For the Question 11, the choices of answers are various combinations of the following:

- \( a = 20 \) or \( a = 30 \)
The distractors are testing novices understanding of the switching of the parameters, which is the difference between answers B and D with the correct answer being A. The sequence of processing the statements provides the difference between answers C and D.

Novices may have been confused between using the values $a$ and $b$ or replacing the values of $x$ and $y$ with $a$ and $b$. Novices may have computed $a = 30$ and $b = 3.0$. However, as for the value of the variable $c$, the answer of $c = 30$ is a strange option to arrive at.

If novices calculate $x \times y$, they will arrive at 90 for the result but if they calculate the value of $a \times b$, they will end up returning the wrong value, of 30. Thus, novices who chose options A and C must be capable of assigning the values to $a$ and $b$.

Option A is the correct answer and distinguishes novices who manage to understand the concept of passing of parameters and are able to analyse the difference between $a, b$ and $x, y$.

Option C returns the value of $\text{int}(a \times b)$, not $\text{int}(x \times y)$ and we classify novices who chose this option as able to apply execute, but this group of novices fails to differentiate between the values of $a, b$ and $x, y$. Options B and D have given the values of $x, y$ to $a, b$. For option B, novices manage to return the value of $c$ correctly, so we classify them as able to apply execute.

But for option D, novices appear to lack understanding of parameter passing, because they have mistakenly given the value of $a, b$ to $x, y$ and have returned $c$ as $\text{int} (a \times b)$ instead of $\text{int} (x \times y)$. To consider D to be the answer requires two incorrect assumptions. So, we classify this response as lack of understanding, while both B and C would involve only one error in assumption, so we classify them as being unable to apply execute, level 3 of our modified version of Bloom’s Taxonomy.

**Question 14**

For Question 14, 34.83% of responses chose the correct answer (option A). 59.55% of responses selected option B as their answer. Novices may have been confused between rows and columns. They are possibly familiar with the template of the multi-dimensional array, but were unable to apply the row and column operations properly, leading to answering option B instead of A. They may be surface learners, as they have quickly recognised and applied something they thought appeared to be familiar, but have not considered their an-
CHAPTER 5. NOVICE RESPONSES TO SUMMATIVE ASSESSMENT

Table 5.7: Analysis of the Novice Responses for Question 14

<table>
<thead>
<tr>
<th>Option</th>
<th>Possible Feedback</th>
<th>Descriptions</th>
<th>Responses in(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 1 2</td>
<td>Analyse - organising</td>
<td>34.83%</td>
</tr>
<tr>
<td></td>
<td>0 1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0 0 0</td>
<td>Able to apply, but surface learning style, leads to error</td>
<td>59.55%</td>
</tr>
<tr>
<td></td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 2 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1 1 1</td>
<td>Lack of understanding</td>
<td>2.25%</td>
</tr>
<tr>
<td></td>
<td>2 2 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 3 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1 0 0</td>
<td>Lack of remembering, leads to wild guess</td>
<td>3.37%</td>
</tr>
<tr>
<td></td>
<td>0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

swer in depth. Option B has involved the confusion of rows and columns which we consider higher order, so have set this at level 4.

There are very few who answered the other options, C and D, which we classify as lacking in levels 2 (understand) and 1 remembering respectively, as we can not see how they might have arrived at the answer given in D, unless they had taken a wild guess.

5.2.5 Classification of Multiple Choice Questions Based on the Modified Version of Bloom’s Taxonomy

In this subsection we extend the classification of the multiple choice questions to all the 16 questions in the summative assessment and present it in Table 5.8. Our classifications of the multiple choice questions are based on the guidelines described in our modified version of Bloom’s Taxonomy (refer Section 5.2.3).

There are three questions in level 2 of our modified version of Bloom’s Taxonomy, eight questions in level 3 and five questions in level 4 of our modified version of Bloom’s Taxonomy. The Programming 1 course requires novices to put their programming concept knowledge into practice. Hence, this explains why most of the questions tested in the multiple choice questions section are classified in these two levels of our modified version of Bloom’s Taxonomy.

77
CHAPTER 5. NOVICE RESPONSES TO SUMMATIVE ASSESSMENT

Table 5.8: Classification of Bloom’s Taxonomy to Multiple Choice Questions

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Bloom’s Taxonomy</th>
<th>Option (in %)</th>
<th>Total (in %)</th>
<th>Correct Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3.37 59.55</td>
<td>34.83 2.25</td>
<td>100 B</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7.87 22.47</td>
<td>61.80 7.87</td>
<td>100 C</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7.87 61.80</td>
<td>20.22 10.11</td>
<td>100 B</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>12.36 16.85</td>
<td>61.80 8.99</td>
<td>100 C</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>12.36 22.47</td>
<td>53.93 11.24</td>
<td>100 C</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>25.84 64.04</td>
<td>6.74 3.37</td>
<td>100 B</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>10.11 20.22</td>
<td>4.49 65.17</td>
<td>100 D</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>34.83 51.69</td>
<td>2.25 11.24</td>
<td>100 B</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>48.31 28.09</td>
<td>15.73 7.87</td>
<td>100 A</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>15.91 17.05</td>
<td>46.59 20.45</td>
<td>100 C</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>34.83 59.55</td>
<td>2.25 3.37</td>
<td>100 A</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>10.11 42.70</td>
<td>12.36 34.83</td>
<td>100 D</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>16.85 26.97</td>
<td>37.08 19.10</td>
<td>100 C</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>1.12 4.49</td>
<td>0.00 94.83</td>
<td>100 D</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>4.49 10.11</td>
<td>17.98 67.42</td>
<td>100 D</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>44.32 2.27</td>
<td>4.55 48.86</td>
<td>100 A</td>
</tr>
</tbody>
</table>

Next, we analyse novices’ correct responses in the specific levels of our modified version of Bloom’s Taxonomy (refer Table 5.9, Table 5.10 and Table 5.11). Questions 1, 17 and 19 are classified in understand level of our modified version of Bloom’s Taxonomy. Only a little number of novices did not manage to identify the correct answer for Question 17, with almost 60% of the responses choosing the correct answer. Question 19 is harder compared to the other questions in this level of our modified version of Bloom’s Taxonomy because only 44.32% of the responses choosing the correct answer.

Table 5.9: Multiple Choice Questions in Level 2 (Understand) of Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Bloom’s Taxonomy</th>
<th>Correct Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>59.55</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>94.83</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>44.32</td>
</tr>
</tbody>
</table>

Questions 2, 3, 4, 5, 6, 8, 9 and 16 are classified in level 3 (apply execute) of our modified version of Bloom’s Taxonomy. In this group of questions, five questions have more than 60% correct responses from novices, two questions have more than 50% correct responses from novices and only Question 16 is not within the range of 50% to 70% of correct responses. Question 16 may be a more difficult question compared to the other questions in this level.
Table 5.10: Multiple Choice Questions in Level 3 (Apply Execute) of Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Bloom’s Taxonomy</th>
<th>Correct Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>61.80</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>61.80</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>61.80</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>53.93</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>64.04</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>65.17</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>51.69</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>37.08</td>
</tr>
</tbody>
</table>

of our modified version of Bloom’s Taxonomy.

Table 5.11: Multiple Choice Questions in Level 4 (Apply Implement) of Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Bloom’s Taxonomy</th>
<th>Correct Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>4</td>
<td>48.31</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>46.59</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>34.83</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>34.83</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>67.42</td>
</tr>
</tbody>
</table>

Questions 11, 12, 14, 15 and 18 are classified in level 4 (apply implement) of our modified version of Bloom’s Taxonomy. In this group of questions, two questions have 34.83% of correct responses and two questions are in the range of 45% to 50% of correct responses. Question 18 is found to be easier compared to the other questions in this level of our modified version of Bloom’s Taxonomy.

Overall, we expected that the lower levels of our modified version of Bloom’s Taxonomy would contain more correct responses from novices compared to the higher levels. This did apply to most of the questions, as we found the pattern that most of the correct responses for the questions in level 4 are in the range of 30% to 50%, level 3 are in the range of 50% to 70% and level 2 are in the range of almost 60% to almost 100%. However, questions 16, 18 and 19 are not in the same range as other questions in the specified level of our modified version of Bloom’s Taxonomy.

Therefore, in order to conform to Scott’s claim in his paper that many students are able to answer at the lower levels of Bloom’s Taxonomy and the chances of them answering correctly decrease as the level of difficulty in the Bloom’s Taxonomy increases [Scott, 2003] we correlate the number of correct responses of each question to the levels of our modified version.
version of Bloom’s Taxonomy.

Table 5.12: Correlations Coefficient ($\rho$) Between Correct Responses and Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Correlations Coefficient Between</th>
<th>N</th>
<th>$\rho$-value</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Responses and Bloom’s Taxonomy</td>
<td>16</td>
<td>-0.334</td>
<td>0.191</td>
</tr>
</tbody>
</table>

We set the $\alpha$ level to 0.05 (refer Section 5.4.6). “N” in the second column of the table represents the number of items tested in the correlation calculation. Referring to Table 5.12, the p-value returned is more than 0.05, therefore there are no significant correlations between the number of correct responses and the levels of Bloom’s Taxonomy.

However, if we remove the three questions that are not in the range for the classified level of modified version of Bloom’s Taxonomy (questions 16, 18 and 19), there exists a strong negative correlation between the number of correct responses and the levels Bloom’s Taxonomy (refer Table 5.13). A negative correlation means as the value of one variable increases, the value of the other variable decreases, and vice versa. As the levels of Bloom’s Taxonomy increase, the lesser the number of students who choose the correct answer. Hence, by removing the three questions, we support Scott’s claim in his paper [Scott, 2003], but there is no significant correlation if we use all the 16 items in the set to test.

Table 5.13: Correlations Coefficient ($\rho$) Between Correct Responses and Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Correlations Coefficient Between</th>
<th>N</th>
<th>$\rho$-value</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Responses and Bloom’s Taxonomy</td>
<td>13</td>
<td>-0.772</td>
<td>0.002</td>
</tr>
</tbody>
</table>

5.3 Reflections

In the first cycle of action research methodology, we analyse the overall and individual responses to a set of multiple choice questions for Programming 1 final examination or summative assessment. Novice abilities to learn programming are not able to be assessed via item analysis. We are not able to identify the classification of the novices’ mental model based on their responses to multiple choice questions. Hence, we classify the novice responses based on our interpretations of the novices’ mental models (on how they attempt to answer a multiple choice question) using our modified version of Bloom’s Taxonomy.

We find that there exists no correlation between the number of correct responses and the levels of the modified version of Bloom’s Taxonomy. However, when we omit three questions (questions 16, 18 and 19) that are not in the range of the classified levels of the modified
version of Bloom’s Taxonomy, there exists a significant negative relationship. This means that the higher level of the modified version of Bloom’s Taxonomy, the lesser the number of correct responses from novices.

In order to analyse the student responses, we discuss three multiple choice questions in our attempt to probe the novices’ mental models. We realise that it is inefficient to define two levels of Bloom’s Taxonomy for the one theme of apply (apply execute and apply implement) because they have the same classification descriptions. Execute and implement are the subcategories for the apply level of Bloom’s Taxonomy. Therefore, we take a step backwards, and explore the classifications of the questions using the original Bloom’s Taxonomy to establish our understandings in applying the Bloom’s Taxonomy in educational research. Hence, in the next section we classify a set of summative assessment based on the original Bloom’s Taxonomy and SOLO Taxonomy.

5.4 Classification of Summative Assessment Based on Learning Taxonomies

In the second cycle of action research methodology we study the questions from summative assessment of semester 1, 2008. In this cycle, we do not replicate the first cycle, but aim for betterment of the research conducted in the first cycle. A total of 220 exam answer sheets were reviewed from the final examinations of semester 1, 2008.

In this section we report the classification of the multiple choice questions based on the original version Bloom’s Taxonomy. We choose to analyse the questions based on the original version of Bloom’s Taxonomy because in Section 5.3, we explained that it is inefficient to use the classification of the modified version of Bloom’s Taxonomy. Hence, we discuss two questions to explain our classification of the questions based on the original Bloom’s Taxonomy. Next, we apply the multiple choice questions to our proposed classifications that involves instructor levels of complexity and novice levels of difficulty. We also report our calculation of the index of discrimination and item analysis to the multiple choice questions.

There are 20 questions in Part 1 of the assessment, but we analyse only 19 multiple choice answers. We omit Question 6 because it requires novices to trace a code segment and cannot be categorised as a multiple choice question since there are no options provided for the answer.

As well, we categorise a short answer coded question based on the SOLO Taxonomy. As explained in Chapter 3, Bloom’s Taxonomy may classify the thinking levels and the SOLO Taxonomy may classify the overall thinking process. Therefore, we try to explore the
expected thinking levels for each of the questions tested in the multiple choice questions and classify the novice responses to a short answer question.

### 5.4.1 Application of Bloom’s Taxonomy to Multiple Choice Questions

In the previous cycle of action research, we conducted overall and individual analyses to novices responses, and then the inadequacy of the analyses lead us to classify the questions using our modified version of Bloom’s Taxonomy. In Section 5.3 we learned that it is inefficient to have the *apply execute* and *apply implement* levels in our modified version of Bloom’s Taxonomy because they have the same classification descriptions. In this cycle, we further our research by examining the multiple choice questions, and the responses and classify them according to the original Bloom’s Taxonomy. We present levels of Bloom’s Taxonomy and the number of questions for each of the level in Table 5.14. The classification for each question based on the Bloom’s Taxonomy is presented in Table 5.17.

#### Table 5.14: Bloom’s Taxonomy and the Number of Multiple Choice Questions in Each Level

<table>
<thead>
<tr>
<th>Level</th>
<th>Category</th>
<th>Number of Multiple Choice Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Evaluate</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Synthesis</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Analysis</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Application</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Comprehension</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>Knowledge</td>
<td>1</td>
</tr>
</tbody>
</table>

We find that the content of the 19 multiple choice questions may be classified primarily into the three lower levels of Bloom’s Taxonomy: *knowledge*, *comprehension* and *application* only. Since this is an introductory programming course, the test instrument should test novices’ performances at the lower level skills. In previous cycle, we had considered that the most of the multiple choice questions in the summative assessment would rate in the *application* level of Bloom’s Taxonomy, however, it turned out that 15 out of the 19 multiple choice questions, are lower and in the *comprehension* level. There is only one question in this test which we categorise as *knowledge*, the lowest level in Bloom’s Taxonomy and three only three questions in *application* level of Bloom’s Taxonomy.

We realise that there are many subcategories for each category presented by the taxonomy, as learning and background information are wide and varied. As such, we find that categorising a novice response is, possibly unsurprisingly, a difficult task which poses some
interesting challenges. Other researches also attempted to classify programming tasks (including reading and code-tracing components) and highlighted that it was difficult to match a programming tasks or questions with Bloom’s cognitive process [Whalley and Robbins, 2007; Whalley et al., 2007]. Firstly, Bloom’s Taxonomy categories were not developed to relate to programming questions. Secondly, it is not obvious which “keyword” in each category to apply to each question.

We also have the difficulty of distinguishing between categorising at the comprehension and application levels in the early stages of learning, and so we decided that, since this is a practical course, novices are expected both to comprehend and to be able to demonstrate the basic knowledge covered at the same time. There are three application level questions in the examination, and these questions require the novices to display their ability to handle an unfamiliar situation such as the one posed in the question.

“A problem in the comprehension category requires the student to know an abstraction well enough that he or she can correctly demonstrate its use when specifically asked to do so. Application, however, requires a step beyond this. Given a problem new to the student, he will apply the appropriate abstraction without having to be prompted as to which abstraction is correct or without having to be shown how to use it in that situation. A demonstration of comprehension shows that the student can use the abstraction when its use is specified. A demonstration of application shows that he will use it correctly, given an appropriate situation in which no mode of solution is specified”. [Bloom and Krathwohl, 1956]

As an example, questions 9 and 15 both require novices to demonstrate their ability to iterate through loops. We present questions 9 and 15 to show the difference between the comprehension and application levels of Bloom’s Taxonomy. The distinction between comprehension and application levels is made by a question at the comprehension level requiring a novice to abstract well enough to demonstrate the expected knowledge. Whereas at the application level, when unfamiliar elements are presented, the novice should be familiar enough with the old existing elements to be able to restructure them and respond correctly to the question [Bloom and Krathwohl, 1956].

Question 9

Question 9 is a “do-while” loop, given the m value of 0, novices are required to count how many times the loop will iterate. This question is classified as a comprehension level in
Bloom’s Taxonomy because novices are expected to demonstrate the understanding that a “do-while” loop must be executed at least once.

9. How many times will the do-while loop below be executed?

```java
int m = 0;
do {
    System.out.println(m);
m = m - 1;
} while (m > 0);
```

A) 0 times  
B) 1 time  
C) 10 times  
D) It is an infinite loop (ie. it will never stop executing)

Question 15

Question 15 requires the novices to understand how the “for-loop” will iterate and present this as a matrix of 3 x 3. The method of presenting the data is a new element for these novices. Thus, in order to solve the question, novices need to have an understanding of the “for-loop” and be able to apply this knowledge as well as present the answer in a matrix. For these reasons we have classified it at the application level of Bloom’s Taxonomy. Question 15 is shown below:

15. What is the output of the program below?

```java
public static void main (String[] args)
{
    int[ ][ ] m = new int[3][3];
    
    for (int row = 0; row <= 2; row++)
    for (int col = 0; col <= 2; col++)
        m[row][col] = col;
    
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<3; j++)
            System.out.print(" "+m[i][j]);
        System.out.println();
    }
}
```
We have provided two questions as example to distinguish between comprehension and application levels of the original Bloom’s Taxonomy. In the next subsections we continue to classify the multiple choice questions in summative assessment based on the instructor viewpoints and the novice responses.

5.4.2 Instructor Levels of Complexity

As a result of identifying the challenges posed by categorisation of questions according to Bloom’s Taxonomy, we investigate the motives of the instructors in setting the questions. We ask the two of the Programming 1 course’s instructors, who were responsible for setting the summative assessment, to provide their assessment of the levels of complexity for each question based on the categorisation we explain in the next paragraph. Both of them discuss and come out with a list of the assessment levels of complexity for each question.

We base our study of the instructor levels of complexity on the depth of the problem posed in the question. We suggest three basic categories to distinguish between the complexity of the questions: low, medium and high. A low complexity represents a very basic coverage of a particular concept, data element or instruction; a medium level is when more than one element is involved but the structure is sequential; and high refers to a question which may involve complex objects or nested structures or combinations of difficult concepts, such as if-statements inside loops. The instructors applied these categories as a basis for judging the
difficulty of each question. We present the explanations, which we gave to instructors, in the survey in Table 5.15.

Table 5.15: Instructor Levels of Complexity

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Definitions, variables, concepts, simple instructions</td>
</tr>
<tr>
<td>Medium</td>
<td>More than one action, object, if and loop statements</td>
</tr>
<tr>
<td>High</td>
<td>Nested structures, complex objects, combinations of difficult concepts</td>
</tr>
</tbody>
</table>

Our classifications for instructor levels of complexity are reported in Table 5.17. The first nine questions analysed were in the low level of complexity determined by instructors. Questions 11 to 14 are medium in complexity, questions 15 to 16 are high in complexity, 17 to 19 medium in complexity and Question 20 is high in difficulty. For this set of questions, we find a pattern that the instructors devise the questions in increasing levels of difficulties from questions 1 to 14 (Question 6 is not included in this range of study because it is not a multiple choice question) and then questions 15 to 20 are mixtures of medium and high levels of complexity given by the programming instructors.

An associated question to consider is how accurate are the instructors’ expectations of complexity. Also, the instructors’ expectations as evidenced by the difficulty of the question may not be on a par with the novices’ performances [Clear et al., 2008; Joni et al., 1983]. We found it worthwhile and interesting not only to look at the complexity level expected by the instructors, but also how this compared with the responses given by the novices, or the novice levels of difficulty.

In order not to confuse the two classifications we have used “complexity” to refer to the instructors’ judgements and “difficulty” to refer to the novices’ responses.

5.4.3 Novice Levels of Difficulty

In our continuing analysis, we look at the level of difficulty faced by the novices. According to the nature of multiple choice questions, they have four possible responses (A, B, C or D), only one correct answer and three distractors. Our scale for describing the novices’ difficulty is based on Lord’s research [Lord, 1952], and is presented in Table 5.16. If there are 85% or more of the novices selecting the correct response, the question is classified as easy. If only 51% to 84% of the novices choose the correct response, this question is of medium difficulty; and if 50% or less select the correct answer, the question is hard for the novices to resolve. Thus the novice levels of difficulty of a question rate as easy, medium or hard.
Table 5.16: Novice Levels of Difficulty

<table>
<thead>
<tr>
<th>Level of Difficulty</th>
<th>Range of correct responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>85 - 100</td>
</tr>
<tr>
<td>Medium</td>
<td>51 - 84</td>
</tr>
<tr>
<td>Hard</td>
<td>0 - 50</td>
</tr>
</tbody>
</table>

Referring to the classifications for the *novice levels of difficulty* as described in Table 5.16, there are five easy multiple choice questions, 10 medium questions and four hard questions in the summative assessment (refer Table 5.17).

Table 5.17: Classifications of Multiple Choice Questions Based on Bloom’s Taxonomy, Instructor Levels of Complexity and Novice Levels of Difficulty

<table>
<thead>
<tr>
<th>QNo</th>
<th>Bloom’s Taxonomy</th>
<th>Instructor Levels of Complexity</th>
<th>Novice Levels of Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Knowledge</td>
<td>Low</td>
<td>Easy</td>
</tr>
<tr>
<td>2</td>
<td>Comprehension</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>3</td>
<td>Comprehension</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>4</td>
<td>Comprehension</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>5</td>
<td>Comprehension</td>
<td>Low</td>
<td>Hard</td>
</tr>
<tr>
<td>7</td>
<td>Application</td>
<td>Low</td>
<td>Easy</td>
</tr>
<tr>
<td>8</td>
<td>Comprehension</td>
<td>Low</td>
<td>Easy</td>
</tr>
<tr>
<td>9</td>
<td>Comprehension</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>10</td>
<td>Comprehension</td>
<td>Low</td>
<td>Hard</td>
</tr>
<tr>
<td>11</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Hard</td>
</tr>
<tr>
<td>12</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>13</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>14</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Easy</td>
</tr>
<tr>
<td>15</td>
<td>Application</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>16</td>
<td>Comprehension</td>
<td>High</td>
<td>Hard</td>
</tr>
<tr>
<td>17</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Easy</td>
</tr>
<tr>
<td>18</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>19</td>
<td>Comprehension</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>20</td>
<td>Application</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>

The overall outcomes of our classifications are presented in Table 5.17. We provide the cognitive levels based on Bloom’s Taxonomy, the instructors’ assessments and novices’ responses for each multiple choice question. Note that “QNo” in the first column refers to the question number. Question 6 is omitted from this table as it was not a multiple choice question.

In the next subsections we further analyse each question by calculating the index of
5.4.4 Index of Discrimination

We have classified the multiple choice questions based on the Bloom’s Taxonomy, and introduced instructor levels of complexity and novice levels of difficulty. We further our investigation of the multiple choice questions by conducting the index of discrimination.

The index of discrimination describes the differences in scoring between different groups of achievers. We divide the novices into three groups for the analysis of the index of discrimination, including: high third of the class, middle third of the class and low third of the class.

These three groups are categorised based on their overall performance in the summative assessment in relation to the rest of the class. Thus, the high third represents the novices who are in the one third of the top scorers in the class. The low third is the one third of the class who scored the lowest, and the Middle third is the one third of the class between the high third and the low third.

In our analysis, we do not emphasise the middle third group but focus on the high third and the low third groups in order to analyse and compare the responses of the high and low scorers. We investigate the questions that are able to distinguish the high and the low scorers. This can be determined by the gap between the high scoring group (high third) and the low scoring group (low third) of novice responses to a particular question. This gap is the index of discrimination.

We sort the result based on the highest and lowest percentage of correctness (refer Table 5.18. “QNo” in the second column represents that question number. The highest percentage of correctness will be ranked as 1 and the lowest will be ranked as 19). Thus, Question 17 with rank 1 is the easiest question, as 96% of the novices chose the correct response. Question 10 is the hardest of the multiple choice questions in the set as only 33% of the responses chose the correct answer.

Question 1 tests at the knowledge level of cognition, instructors classify it in the low level of complexity and novices found it was easy in terms of its difficulty, but referring to Table 5.18, Question 1 is not the easiest among the set of multiple choice questions as it is ranked number 4.

The largest gap is in the responses between the high third group and low third group occurred for Question 19, with a 67% difference (refer Table 5.18). Question 19 tests on a
“while loop” and an “array”, and requires novices to select the answer with the values of the elements in the array when the loop finishes. This question is in the set of the Bracelet’s group questions. There were 93% of the high third group and only 26% of the low third group who managed to select the correct answer to this question. This question could be considered as a good predictor to distinguish the top and low achievers as it has the largest gap between the high third group and low third group from the set of multiple choice questions.

The lowest gap occurs for questions 8 and 17, with a 14% difference. These questions appear to be the top two in the rank, which means they are the easiest questions from the set. For Question 17, the easiest in the set, there were 86% of correct responses from the low third group and 100% of correct responses from the high third group. For Question 8, there were 85% of correct responses from the low third group and 99% of correct responses from the high third group.

A closer investigation identifies the hardest question, the lowest in the rank (refer Table 5.18), as Question 10. It reveals that the novices found it to be the hardest of all the multiple choice questions, with 41% in the index of discrimination but the instructors categorised this question as a low of complexity (refer Table 5.17).

5.4.5 Item Analysis

We further narrow our investigation by conducting item analysis [Levesque, 1998] on the responses for each option provided in the multiple choice questions for the novices based on their three groups; high third, middle third and low third (see Section 5.4.4). In this cycle, the item analysis refers to a question analysis which is conducted to highlight a few questions and attempt to justify the rationale of each option chosen by the novices. We accept that there may be educated guesses in answering the multiple choice questions, but as a whole, novices will have to think and plan a strategy before choosing any option.

In Section 5.4.4 we identified the hardest question, the lowest in the rank of Table 5.18, as Question 10. Question 10 is classified in the comprehension level of Bloom’s Taxonomy and is low in complexity but found to be the hardest (ranked number 19) in the set of multiple choice questions, with 41% in the index of discrimination. Question 10 is a question to test novices’ comprehension about object-oriented concepts (refer Appendix C.3). This is a theory-based question, with the statements which could be true or false, and requires novices to identify which statement about static and non-static methods is false. The novices are required to distinguish the false statement from the given choices which alone makes the
Table 5.18: Index of Discrimination (ID)

<table>
<thead>
<tr>
<th>Rank</th>
<th>QNo</th>
<th>Correct Responses (in %)</th>
<th>High third</th>
<th>Middle third</th>
<th>Low third</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>96</td>
<td>100</td>
<td>99</td>
<td>86</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>93</td>
<td>99</td>
<td>95</td>
<td>85</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>87</td>
<td>96</td>
<td>9</td>
<td>73</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>85</td>
<td>99</td>
<td>87</td>
<td>68</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>80</td>
<td>97</td>
<td>80</td>
<td>58</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>71</td>
<td>92</td>
<td>76</td>
<td>39</td>
<td>52</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>70</td>
<td>90</td>
<td>77</td>
<td>39</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>70</td>
<td>92</td>
<td>70</td>
<td>45</td>
<td>46</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>69</td>
<td>93</td>
<td>70</td>
<td>38</td>
<td>55</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>67</td>
<td>96</td>
<td>67</td>
<td>33</td>
<td>63</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>66</td>
<td>88</td>
<td>67</td>
<td>41</td>
<td>47</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>65</td>
<td>89</td>
<td>67</td>
<td>33</td>
<td>56</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>64</td>
<td>81</td>
<td>63</td>
<td>45</td>
<td>35</td>
</tr>
<tr>
<td>14</td>
<td>19</td>
<td>63</td>
<td>93</td>
<td>65</td>
<td>26</td>
<td>67</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>62</td>
<td>83</td>
<td>56</td>
<td>47</td>
<td>36</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>50</td>
<td>85</td>
<td>41</td>
<td>20</td>
<td>65</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>40</td>
<td>58</td>
<td>43</td>
<td>17</td>
<td>41</td>
</tr>
<tr>
<td>18</td>
<td>16</td>
<td>35</td>
<td>65</td>
<td>27</td>
<td>12</td>
<td>53</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>33</td>
<td>56</td>
<td>27</td>
<td>16</td>
<td>41</td>
</tr>
</tbody>
</table>

question difficult and confusing. We conduct an item analysis to consider the division of responses for each of the options A, B, C and D for Question 10 and present our analysis in Table 5.19.

Table 5.19: Item Analysis on Question 10

<table>
<thead>
<tr>
<th>Group within class</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>High third</td>
<td>8.5</td>
<td>56.3</td>
<td>19.7</td>
<td>15.5</td>
<td>100.0</td>
</tr>
<tr>
<td>Middle third</td>
<td>9.8</td>
<td>26.8</td>
<td>28.0</td>
<td>35.4</td>
<td>100.0</td>
</tr>
<tr>
<td>Low third</td>
<td>28.1</td>
<td>15.6</td>
<td>26.6</td>
<td>29.7</td>
<td>100.0</td>
</tr>
<tr>
<td>Total overall</td>
<td>14.7</td>
<td>33.2</td>
<td>24.9</td>
<td>27.2</td>
<td>100.0</td>
</tr>
</tbody>
</table>

The correct answer to question 10 is option B. More than half of the high third group of the class were able to choose the correct answer. However, within the middle third and low third groups, 35.4% and 29.7% respectively chose distractor D which states that “All objects created from a class which includes a static variable may change the value stored in that static variable (visibility permitting)”. It would appear that a particular group of
novices were guessing or confused or both. It can be argued that this sort of question may be included for variety, and to identify the stronger novices.

Question 11 is in rank number seventeen of Table 5.18, with the same percentage as Question 10 for the index of discrimination. Table 5.20 reports the division of responses for each of the options A, B, C and D for Question 11. Question 11 is a hard question on the novice levels of difficulty scale and has been classified at the comprehension level of Bloom’s Taxonomy, medium level of complexity by the instructors and 41% in the index of discrimination.

<table>
<thead>
<tr>
<th>Group within class</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>High third</td>
<td>22.2</td>
<td>58.3</td>
<td>19.4</td>
<td>0</td>
<td>100.0</td>
</tr>
<tr>
<td>Middle third</td>
<td>7.3</td>
<td>42.7</td>
<td>41.5</td>
<td>8.5</td>
<td>100.0</td>
</tr>
<tr>
<td>Low third</td>
<td>9.4</td>
<td>17.2</td>
<td>68.8</td>
<td>4.7</td>
<td>100.0</td>
</tr>
<tr>
<td>Total overall</td>
<td>12.8</td>
<td>40.4</td>
<td>42.2</td>
<td>4.6</td>
<td>100.0</td>
</tr>
</tbody>
</table>

The correct answer for Question 11 is option B and more than half of the high third group manage to answer the question correctly. There are 58.3% of responses from the high third group choosing the correct answer. However, it would appear that only 17.2% of responses from the low third group understand parameter passing. The majority, 68.8% of the low third group have selected the distractor C, which follows an incorrect path for passing parameters from a public method to main of the program.

The question in rank one is Question 17 (refer Table 5.18), which is in the comprehension level of Bloom’s Taxonomy and found easy by the novices but medium in complexity by the instructors. This question tests on the “if-statement” with a complex test condition, and all of the high third group, 98.8% middle third group and 86.4% of the low third group chose the correct answer, which brings to the lowest index of discrimination of the set of multiple choice questions to 14%. The following item analysis shows the slips in their choices (refer Table 5.21).

<table>
<thead>
<tr>
<th>Group within class</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>High third</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.0</td>
</tr>
<tr>
<td>Middle third</td>
<td>98.8</td>
<td>1.2</td>
<td>0</td>
<td>0</td>
<td>100.0</td>
</tr>
<tr>
<td>Low third</td>
<td>86.4</td>
<td>7.6</td>
<td>0</td>
<td>6.1</td>
<td>100.0</td>
</tr>
<tr>
<td>Total overall</td>
<td>95.5</td>
<td>2.7</td>
<td>0</td>
<td>1.8</td>
<td>100.0</td>
</tr>
</tbody>
</table>
CHAPTER 5. NOVICE RESPONSES TO SUMMATIVE ASSESSMENT

All the novices in the high third group and almost all the novices in the middle third group chose the correct answer for this question. The answer for Question 17 is option A. The distractors B and D, the second and third output, have the same pattern of the answer, “Success! Failure!”, which is the total opposite of the correct answer. One possible explanation for this slip is that novices may take a glance at the question and assume the value of the variable $x$ to be $0<x<50$, instead of $(x>50 && x>0)$, which is further confirmed by the choice of distractor B. Most probably, 1.8% of all the novices from the lowest level group have guessed option D or they were unable to apply any of their knowledge to this question. Question 17 is tagged as the easiest question at the comprehension level of Bloom’s Taxonomy (Table 5.21) and a total of 95.5% of the novices chose the correct answer.

Question 5 has the same content pattern as Question 17, an “if-statement”, but Question 5 has two “if-statements” straight after each other. Both questions test on the comprehension level of the novices. Novices found Question 17 as easy, but Question 5 as hard.

<table>
<thead>
<tr>
<th>Group within class</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>High third</td>
<td>15.3</td>
<td>0</td>
<td>84.7</td>
<td>0</td>
<td>100.0</td>
</tr>
<tr>
<td>Middle third</td>
<td>51.2</td>
<td>0</td>
<td>41.5</td>
<td>7.3</td>
<td>100.0</td>
</tr>
<tr>
<td>Low third</td>
<td>60.6</td>
<td>0</td>
<td>19.7</td>
<td>19.7</td>
<td>100.0</td>
</tr>
<tr>
<td>Total overall</td>
<td>42.3</td>
<td>0</td>
<td>49.1</td>
<td>8.6</td>
<td>100.0</td>
</tr>
</tbody>
</table>

The correct answer for this question is option C (Table 5.22). The majority of the high third group managed to select the correct answer. The discrimination index for this question is 65%, which shows a large gap between the high third group and the low third group, despite the instructors considering it a low level of complexity.

More than half of the responses in the low third and middle third groups selected distractor A. They were possibly not aware that this snippet code is not really one “nested if-else”, instead it is two “if-else statements”. By applying their superficial “nested if” knowledge, they have confidently chosen option A, without going through the whole code. Attention to the detail of code is perhaps a higher level of cognition which is being tested in this question.

For Question 5, 50.9% of the novices were unable to successfully trace the code through the “nested if-else” statement. This topic is covered in class. However, the level of difficulty of the “if-statement” is largely unrecognised. In fact they require novices to retain or remember one option as they compare each alternative down one path, and then come back to the conclusion. One suggestion or recommendation for novices to tackle such questions is to draw a diagram.
to help them with tracing the variables and their values down the paths. The cognitive level in the Bloom’s Taxonomy for the conditions or branching is not categorised because this taxonomy existed long before it was applied to programming tasks, but it is generally considered to be at the *comprehension* level of the concept of sequence and the novices may be unfamiliar to the pattern of sequential “if” statements.

Question 16 is *hard* and at a *high* level of complexity and in the *comprehension* level of Bloom’s Taxonomy. Only 35% of the novices selected the correct answer, with a 53% score in the index of discrimination.

Table 5.23: Item Analysis on Question 16

<table>
<thead>
<tr>
<th>Group within class</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>High third</td>
<td>15.3</td>
<td>65.3</td>
<td>15.3</td>
<td>4.2</td>
<td>100.0</td>
</tr>
<tr>
<td>Middle third</td>
<td>13.4</td>
<td>26.8</td>
<td>41.5</td>
<td>18.3</td>
<td>100.0</td>
</tr>
<tr>
<td>Low third</td>
<td>4.5</td>
<td>12.1</td>
<td>43.9</td>
<td>39.4</td>
<td>100.0</td>
</tr>
<tr>
<td>Total</td>
<td>11.4</td>
<td>35.0</td>
<td>33.6</td>
<td>20.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

The topic of exceptions has been covered towards the end of the semester which means the instructors recognise it as a difficult area. The answer for Question 16 is option B. 65.3% of the *high third* group selected the correct answer (Table 5.23) to this question covering exceptions. There were 33.6% of the responses which selected distractor C. These novices either did not know or forgot the rule of thumb for the “exception” which is that some paths may not be compiled or ever reached. Most probably because there is no “throw new ExceptionTypeTwo” instruction, the novices simply move on to the next line of code which leads them to select distractor C. As for the 20% responses who chose distractor D, they possibly considered that the code would “throw new ExceptionTypeOne” instead of “throw new ExceptionTypeTwo”. Only 11.4% of the responses chose distractor A which portrays that this group of novices did not understand the concept of an “exception” at all, thus preventing them from applying it.

Since there are differences in our applications to the classifications of Bloom’s Taxonomy, *instructor levels of complexity*, *novice levels of difficulty* and index of discrimination, in the next subsection we correlate these four classifications. The rankings of the analysis from index of discrimination may agree with the *novice levels of difficulty* as both were analysed based on the novices’ correct responses.
5.4.6 Data Analysis

We analyse the correlation between Bloom’s Taxonomy and the instructor levels of complexity, Bloom’s Taxonomy and the novice levels of difficulty, Bloom’s Taxonomy and the index of discrimination, instructor levels of complexity and novice levels of difficulty instructor levels of complexity and the index of discrimination and, novice levels of difficulty and the index of discrimination. We have explained the details regarding correlations in Section 4.12.1. The results of the correlations are presented in Table 5.24.

Table 5.24: Correlations Coefficient ($\rho$) Between Pairwise Bloom’s Taxonomy, Instructor Level of Complexity (Instructor), Novice Level of Difficulty (Novice) and Index of Discrimination

<table>
<thead>
<tr>
<th>Correlations Coefficient Between</th>
<th>$\rho$-value</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bloom’s Taxonomy and Instructor</td>
<td>0.425</td>
<td>0.070</td>
</tr>
<tr>
<td>Bloom’s Taxonomy and Novice</td>
<td>0.018</td>
<td>0.941</td>
</tr>
<tr>
<td>Bloom’s Taxonomy and Index of Discrimination</td>
<td>-0.056</td>
<td>0.819</td>
</tr>
<tr>
<td>Instructor and Novice</td>
<td>0.177</td>
<td>0.468</td>
</tr>
<tr>
<td>Instructor and Index of Discrimination</td>
<td>0.056</td>
<td>0.820</td>
</tr>
<tr>
<td>Novice and Index of Discrimination</td>
<td>0.617</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Since the P-value for the correlation coefficient between Bloom’s Taxonomy and the instructor levels of complexity is 0.070, which is greater than 0.05, we can say that there is no significant correlation between Bloom’s Taxonomy and instructor levels of complexity at the confidence level of $\alpha = 0.05$.

Similarly for the analysis of correlations between Bloom’s Taxonomy and the novice levels of difficulty, the P-value of 0.941 is greater than 0.05, and between Bloom’s Taxonomy and the index of discrimination, the P-value of 0.819 is greater than 0.05, and between the instructor levels of complexity and the novice level of difficulty, the P-value is 0.468 which is much greater than 0.05, and between the instructor levels of complexity and the index of discrimination, the P-value of 0.820 is greater than 0.05 indicating that that are no significant correlations between the variables mentioned.

Hence, we can say that there are no significant correlations between:

- Bloom’s Taxonomy and instructor levels of complexity
- Bloom’s Taxonomy and novice levels of difficulty
- Bloom’s Taxonomy and index of discrimination
Novices may have different views in terms of the difficulty compared to instructor perspectives of complexity of the questions tested in this summative assessment. As an example, instructors rated questions one to ten in the summative assessment as low in complexity, but novices found that the difficulty of the questions varied based on their responses. Hence, we cannot conclude that if a question is low in complexity, that a novice will find it easy to answer correctly.

There exists a moderate correlation between novice levels of difficulty and the index of discrimination. The values for the index of discrimination were derived from the novices’ correct responses which we have classified based on the novice levels of difficulty. The distributions of the gap between the high third and low third may be influenced by the overall performance of the correct responses of a particular questions which we define as novice levels of difficulty. Meaning to say, if a question has a high number of correct responses (or is easy in novice levels of difficulty), it is likely the gap between the high third and low third of the index of discrimination will be low.

5.4.7 Categorisation of Code Writing Question

We extend our study to analyse Question 24, a short answer question in Part 2 of the summative assessment. We choose this question as it requires short and precise responses. There is a clear instruction provided to the novices. For this question novices are expected to write programming code to calculate the highest and lowest integer (number) from a set of integers passed via the command-line. Question 24 is shown as below:

Complete the HighLow class below to identify and display the highest and lowest of the series of positive integer values passed into the program as command line arguments.

Expected Input/Output is shown below.

%java HighLow 7 4 9 10
Highest value passed in was 10
Lowest value passed in was 4
% java HighLow 45 52 81 69 23 97 76
Highest value passed in was 97
Lowest value passed in was 23

Notes: Command-line arguments are passed to the main method through the array of String references (args in the main method below). The size of any array can be accessed through its length attribute (note: you can assume at least one valid argument will be passed in on the command line). You will need to use Integer.parseInt() to convert each command line argument to integer format before processing it.

(5 marks)

public class HighLow
{
    public static void main (String[] args)
    {
        int highestArg = 0;
        int lowestArg = 0;
        int nextArg;
        System.out.println( "Highest value passed in was " + highest);
        System.out.println( "Lowest value passed in was " + lowest);
    }
}

We are not able to classify the question using Bloom’s Taxonomy because the response to this question is subjective and we wanted to study the interconnection of the cognitive processes from novices’ responses. The SOLO Taxonomy has been used to classify program code [Lister et al., 2006]. We find the SOLO Taxonomy is more suitable compared to Bloom’s Taxonomy to classify novices’ answers for Question 24.

Based on the SOLO Taxonomy, the observed learning outcomes for this question are the skills to create a loop to compare all the numbers and to determine the highest and lowest
value from the range of numbers given. We have slightly modified the categories to enable us to categorise all the responses. We have added the last category, *no attempt or totally wrong*. Our modified set of SOLO Taxonomy categories is presented in Table 5.25.

When classifying according to the SOLO Taxonomy, we investigate a particular solution as one whole model and then consider how the meta-cognitions may have formed. Thus, we find that it is important to analyse how each component of the solution has been coded to build a *relational* level of solution. We study the interrelation of each component in order to understand whether the novice could link all the components which can be represented by their abilities in programming.

For example, we outline a few components that should contribute to the *relational* model in answering Question 24. The components are:

1. Ability to create a loop
2. Ability to find the highest value
3. Ability to find the lowest value
4. Ability to extract or convert the argument correctly
5. Ability to code correctly

We used this approach to distinguish between *relational*, *multistructural*, *unistructural* and *prestructural* models. Item 1 (ability to create a loop) or 2 (ability to find the highest value) or 3 (ability to find the lowest value) is *unistructural*, item 4 (ability to extract or convert the argument correctly) is *multistructural* and item 5 (ability to code correctly) is *relational*. We felt that the extended abstract level could not be tested in this question as the novices had been given clear instructions about which question to solve.

5.4.8 Application of SOLO Taxonomy to Short Answer Question

We applied the SOLO Taxonomy to the responses for Question 24, which required novices to write a code segment about an array to determine the highest and the lowest integer values.

We found that 35.6% of the novice answers were at the *relational* level, 23.3% of responses were at the *multistructural* level and less than 25% were at the lower level categories of SOLO Taxonomy. There were 17.4% responses that had no attempt to answer the question or gave an answer which portrayed that they did not have the correct mental model to respond to the question.
Table 5.25: Category and Descriptions of SOLO Taxonomy (Improvised based on [Biggs, 2008a;b; James, 2011])

<table>
<thead>
<tr>
<th>Category</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Abstract</td>
<td>Novices able to make connections beyond the scope of question and able to transfer knowledge a new situation.</td>
</tr>
<tr>
<td>Relational</td>
<td>Fully correct or almost right. Novices appreciate significance in relation to the whole program and can generalise outside of program.</td>
</tr>
<tr>
<td>Multistructural</td>
<td>There are numbers of connections made. Novices can create code for loops and comparisons, but there are a few minor slips, leading to failure to connect the whole idea. They may fail to convert arguments, use incorrect operators or not interpret general explanation.</td>
</tr>
<tr>
<td>Unistructural</td>
<td>Simple connections are made. Novices can compare, or write loops but fail to implement or derive the connections of loops in relation to manipulation of arrays or usage of further structures.</td>
</tr>
<tr>
<td>Prestructural</td>
<td>There are bits of unconnected information. Novices know something, but the overall argument makes no sense.</td>
</tr>
<tr>
<td>No attempt or totally wrong</td>
<td>The answer is blank or totally wrong.</td>
</tr>
</tbody>
</table>

In the next section we provide examples of novice responses in the summative assessment categorised using the SOLO Taxonomy.

Sample of Relational Model Response

In this category novices were able to use their skills to create a loop and compare between the integer values passed into the program with the correct integer types. An example of a novice response in this category is shown below.

```java
for (int i = 0; i < args.length; i++)
{
    nextArg = Integer.parseInt(args[i]);
}
```
CHAPTER 5. NOVICE RESPONSES TO SUMMATIVE ASSESSMENT

Table 5.26: SOLO Taxonomy Applied to Responses for the Short Written Code Segment

<table>
<thead>
<tr>
<th>Category</th>
<th>Novices’ Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>35.6</td>
</tr>
<tr>
<td>Multistructural</td>
<td>23.3</td>
</tr>
<tr>
<td>Unistructural</td>
<td>16.0</td>
</tr>
<tr>
<td>Prestructural</td>
<td>7.8</td>
</tr>
<tr>
<td>No attempt or totally wrong</td>
<td>17.4</td>
</tr>
</tbody>
</table>

```java
if (nextArg > highestArg)
    highestArg = nextArg;

if (nextArg < lowestArg)
    lowestArg = nextArg;
```

Sample of Multistructural Model Response

In this category novices were able to create loops and compare integers, but are unable to present a working program. In this example the novice did not convert the argument.

```java
for (int i = 0; i < args.length; i++)
{
    if (nextArg > highestArg)
        highestArg = nextArg;

    if (nextArg < lowestArg)
        lowestArg = nextArg;
}
```

Sample of Unistructural Model Response

In the sample of this category novices were able to compare the integers but not able to create a loop to manipulate the array.

```java
if (nextArg > highestArg)
    highestArg = nextArg;

if (nextArg < lowestArg)
    lowestArg = nextArg;
```
Sample of Prestructural Model Response

In the sample of this category novices had the idea to compare the arguments but failed to present it according to the specification of the question and used predefined functions to obtain the maximum and minimum numbers.

\[
\text{min}(\text{lowestArg}, \text{highestArg}); \\
\text{max}(\text{lowestArg}, \text{highestArg});
\]

Example of No Attempt or Totally Wrong Model Response

In the sample of this category the novices made no attempt to answer the question. Some novices made an attempt to answer this question by only rewriting the variable name.

5.5 Discussion

In cycle one of action research methodology, we analysed overall and individual responses to the summative assessment for Programming 1 in semester 2, 2007. We classified the multiple choice questions based on Bloom’s Taxonomy. We discussed our reflections in Section 5.3 that would lead to improvements in analysing the set of questions from the summative assessment for Programming 1 in semester 1, 2008.

In the second cycle of action research methodology, we analysed the multiple choice questions using the original version of Bloom’s Taxonomy. We introduced the instructor levels of complexity and novice levels of difficulty measures for the exam questions. Then, we conducted an index of discrimination and item analysis to analyse the novice responses for each question and optional answer.

The Bloom’s Taxonomy describes the level of cognitive difficulty in learning. We found that it is very difficult to categorise the correct answer, let alone the other multiple options provided in each of the questions. The Bloom’s Taxonomy’s categories were not developed to relate to programming questions. Plus, it was not obvious which “keyword” in each category to apply to each question. In particular, we have difficulty in distinguishing between categorising the questions according to the comprehension or the application levels. At first, we thought that any question that includes written code was devised to test the novices’ ability at the application level of Bloom’s Taxonomy, but we discovered that it is not determined by the question itself, it is about how the novices perceived the questions. We decided that the
CHAPTER 5. NOVICE RESPONSES TO SUMMATIVE ASSESSMENT

comprehension and application levels could be distinguished by the amount of abstraction required. If a novice needs to abstract in order to solve a question, then the question is developed to test the application level of Bloom’s Taxonomy, whereas, if a novice does not need to abstract, then the question is developed to test the comprehension level of Bloom’s Taxonomy.

We also discovered that the three highest categories, analysis (Level 4), synthesis (Level 5) and evaluation (Level 6), were not tested by this set of multiple choice questions for novice programmers. These three levels were tested and seen in the programming project conducted during the semester, wherein novices were able to practise their generating, planning and production skills and also to evaluate their fellow classmates’ projects or compare others’ code with their own.

We further analysed the instructor levels of complexity as we wanted to understand their expectations when they devised the questions. We provided a simple classification as a guideline for them to categorise the questions. We understand that the instructors may have strong background knowledge of programming, and thus tend to find the questions less complex than novices do. We also described the novice levels of difficulty to classify novice responses to the multiple choice questions.

We found that in the set of multiple choice questions, novices may have different views in terms of the difficulty compared to instructor perspectives of complexity of the questions tested in the summative assessments. We admit that a few questions may portray that the three classifications may agree with each other, but overall, our analysis based on the correlation coefficient test showed that the three variables (Bloom’s Taxonomy, instructor levels of complexity and novice levels of difficulty) are not related to each other. We are not able to justify why the three variables are independent, but we may study this issue in future research.

There were a few tricky questions presented in the set of multiple choice questions. These sorts of questions can be justified by suggesting that testing the novices with code which could well have been written by other novices is important because they need to know how to deal with such code and learn from it. In term of complexity, a tricky question may be regarded as a higher level compared to other questions with the same characteristics of classification.

As well as the results presented in Section 5.4.7, we identify the “plain-language” group, which respond to the question, not in Java language, but using English language to solve the question. We observe that, though some novices may have demonstrated strong problem solving skills, they lacked the necessary programming syntax knowledge and are consequently
unable to write any code but could describe in words how to solve this question correctly.

In categorising the answers, we found that on some level the novices may have the logic to answer the question, but due to the lack of syntax knowledge they are not confident to write the answer since the question explicitly asked for code.

We found that there is a small group of novices, 0.9% of the group, in the multistructural level, who may have strong problem solving skills but are unable to write any code, though they could describe in words how to solve this question correctly. We categorise this answer as multistructural as they have all the ability in the components as mentioned in Section 5.4.7 but they lack the necessary syntax knowledge.

However, we argue that the process of classifying the responses based on the SOLO Taxonomy is not easy either. Although we provide guidelines to do so, the interpretation of human responses is very subjective. As an example, the sample of the multistructural model response (refer to Section 5.4.8) can also be classified as a response in the relational model, but with errors. There are distinctions that are not clear in classifying the responses. This area is subjective and each classification needs proper guidelines with solid arguments.

In analysing the responses to the short answer question, we realised that the emphasis of the question was on the correct grammatical syntax and no marks were awarded for having the correct logic or necessary problem solving skills. Therefore, a strong syntax knowledge requirement is paramount to passing such assessment. If novices are unable to practise or write their answers using the correct syntax, they may be discouraged from proceeding further towards answering this type of question. We concur with other studies that have employed the SOLO Taxonomy, that, find it useful to evaluate responses of novices learning to read programs [Thompson, 2007; Whalley et al., 2006; Lister et al., 2006] to evaluate based on novices interpretation of a program. In summative assessment, the instructor should also award marks for the problem solving abilities shown in students’ answers.

5.6 Summary

In this chapter we conclude that each question is unique and needs proper weighting (or scoring) and classification in order to test the specified levels of skills in programming to meet the learning goals. We conducted overall responses, individual responses, item analysis and calculated the index of discrimination to identify the questions that may represent multiple level of difficulties. We applied existing taxonomies, such as Bloom’s Taxonomy to ideally classify the multiple choice questions in the summative assessment in terms of novices’ abili-
ties required to solve the question, but inadequately describe the level of complexity and its difficulty. Since this is for an introductory programming course, the questions on our summative assessment belonged to the lower levels of Bloom’s Taxonomy. But even then, the questions posed at the knowledge and comprehension levels can be difficult for some individuals. Although the content may be an “if statement”, for instance, we found that depending on the nested alternatives, or the complexity of the test condition, that this can add other levels of difficulty to the question. Hence, we incorporated additional classifications to the existing Bloom’s Taxonomies to explain the instructor levels of complexity and the novice levels of difficulty. We found that the combination of these three classifications allowed us to more clearly classify the questions and responses of the summative assessment.

In addition, we recommend the SOLO Taxonomy to classify the novices’ understanding of the particular concepts tested. The SOLO Taxonomy provides a means of evaluating cognitive or mental models to see if the novices are able to make connections between what they have learnt.

Thus, for classifying multiple choice questions of summative assessment, we recommend the three classifications consisting of Bloom’s Taxonomy categories for the cognitive levels tested, the instructor estimates of complexity (instructor levels of complexity) and the novice percentages of correct responses as measures to classify the levels of difficulty (novice levels of difficulty). We recommend the two additional classifications as we have seen that questions exist which are low level in complexity as determined by the instructors but the novices found them difficult to solve.

The next chapter presents a further investigation to inform us how to bridge the gap between the rating of questions by instructors and, novices, in order to better understand the instructors’ expectations in terms of complexity and the novices’ difficulties in programming. Our next goal is to minimise this gap so that both novices and instructors may work together to overcome some of the learning difficulties faced by novices.
Chapter 6

Instructor Perspectives of Multiple Choice Questions

In academia students depend on instructors to guide them to learn. In the previous chapter we analysed novice responses to multiple choice questions in summative assessment for the introductory programming course. We observed that most of the multiple choice questions were at the lower levels of Bloom’s Taxonomy (knowledge and comprehension and application), but not all students found them as easy to answer. We proposed new classifications, novice levels of difficulty and instructor levels of complexity, to establish the variability of the cognitive levels posed by the Bloom’s Taxonomy.

In this chapter we further our investigation of instructor perspectives by conducting a survey to obtain instructor insights about multiple choice questions employed in summative assessment, in introductory programming courses. We hope to improve our classification proposed (instructor levels of complexity) in Chapter 5.

We outline this chapter as follows. Section 6.1 explains our motivation for conducting this part of our research. In Section 6.2 we design and develop the survey. Section 6.3 explains the criteria of instructor levels of complexity as a guide for instructors to classify the multiple choice questions. We aggregate and present our survey data in Section 6.4, our analysis thereof in Section 6.5 and discussions in Section 6.6.
6.1 Motivation: Instructor Insights into Multiple Choice Questions in Summative Assessments

In Chapter 5 we classified two sets of multiple choice questions in summative assessments. As shown in Table 5.17, a question in a comprehension level of Bloom’s Taxonomy can be classified as low in complexity by instructors but novices found it to be hard in terms of its difficulty. This shows that instructor and novice views towards multiple choice questions tested in assessment may vary in terms of the perceived degrees of complexity.

We continue our research from the previous cycle of action research and focus on instructor views of summative assessment, specifically those that employ multiple choice questions. Kinnunen et al. mentioned that an alternative way to explore student learning is to study instructor insights rather than student views [Kinnunen et al., 2007]. We hope that a better understanding of instructor views about assessment will contribute towards research into a more informed understanding of novice programmer difficulties (as discussed in Section 2.2.1).

In Section 5.4.2, we suggested a classification of the instructor levels of complexity. We discussed the classifications based on the complexity of the task, that do not involve the descriptions of the abilities tested for the questions. Hence, in this chapter we will further improve the instructor levels of complexity by proposing criteria that reflect the abilities required for novices to program.

In this chapter, we seek to answer the following questions:

- Do instructors think that summative assessment provides a valid measure of a student’s ability to program?

- What are instructor perceptions of multiple choice questions?

- Are there any correlations between these variables:
  - instructor experiences in teaching, and
  - devising final examination questions, and
  - instructor evaluations of questions.

6.2 Instructor Survey

Our aim is to investigate and explain the instructor perspectives of summative assessment, specifically, when instructors use multiple choice questions to test novice-level programming
CHAPTER 6. INSTRUCTOR PERSPECTIVES OF MULTIPLE CHOICE QUESTIONS

abilities. The survey was made available online and so was easily accessed by potential respondents from all over the world. The complete survey is in Appendix D, with the broad design thereof presented in Figure 6.1.

![Outline for Instructor Survey](image)

**Figure 6.1: Outline for Instructor Survey**

The survey contain two parts (Section A and B) comprising a total of 11 questions (refer Figure 6.1), the questions being denoted as Q1 for Question 1, Q2 for Question 2, and so on.

There are seven questions in Section A and four questions in Section B. Note that the dotted line is to separate the questions in Section A and Section B, which is not clear from the layout alone. Section A seek instructor backgrounds and perspectives of summative assessment. Q1 and Q2 cover the instructor backgrounds, Q6 and Q7 cover instructor perspectives regarding summative assessment, Q3 and Q4 cover instructor views of multiple choice questions and Q5 covers specifically regarding distractors of the multiple choice questions. The four questions in Section B extract instructor evaluation of particular questions.

The survey was open to instructors of a range of programming courses, including instructors of courses variably named at our institution as *Introduction to Programming, Programming 1, Programming 2, Programming 3* and *Java for Programmers*. Further descriptions of
these courses are in Table 4.7.

In the next subsection we provide the details of each question in the survey.

6.2.1 Section A

Section A of the survey consists of seven questions that cover instructor backgrounds and instructor views of summative assessment, multiple choice questions as a test instrument in summative assessment, and their opinions about creating distractors for multiple choice questions (as outlined in Figure 6.1).

Question 1 and Question 2 ask for the instructor teaching backgrounds (including their teaching experience measured in semesters and the courses taught) and the number of semesters during which they have devised questions for summative assessments. This is to establish the experience of the instructors in terms of teaching and assessment preparation for programming courses.

Question 1

In Question 1 we seek to investigate the teaching experience of instructors, including lecturing, teaching and tutoring of programming courses. In addition, we would like to know about their experiences in devising questions for final examinations (which we refer to as summative assessments). Apart from getting to know their backgrounds, we also are keen to determine if instructors would rate the questions in Section B differently, based on their experience in programming courses.

Question 1: For how many semesters you have been conducting the following teaching duties?

a) lecturing / teaching / tutoring programming
b) devising questions for programming examinations

Question 2

In Question 2, we determine which courses the instructors have been involved in, either teaching or devising examination questions, or both. We also describe the categorisation of the courses.

We distribute the survey to instructors who are involved in Introduction to Programming and Programming 1 (these two are fundamental programming courses), and Programming 2
and Programming 3 (these two are intermediate programming courses) to get the insights of programming instructors about summative assessment and multiple choice questions. The course name may be listed differently in other institution or university, and therefore we provide the course descriptions for Introduction to Programming, Programming 1, Programming 2 and Programming 3 on the survey form.

**Question 2: Which level of Programming courses have you covered?**
(You may select more than one response)

- **Introduction to Programming**: This course covers foundations of computer systems, conceptual building blocks for programming, typically for non-Computer Science or IT related majors. (Zero prerequisite)

- **Programming 1**: This course covers introductory programming concepts for novices. (Introduction to Programming or Zero prerequisite)

- **Programming 2**: This course extends the learning of programming to cover more difficult programming principles. (Programming 1 course as prerequisite)

- **Programming 3**: This course covers a detailed study of programming; including the use of defensive programming, debugging, testing, coding standards and practices. (Programming 1 and Programming 2 courses as prerequisites)

**Question 3**

Question 3 requires instructors to provide the reasons why they would use multiple choice questions in summative assessment. We designed a five-point Likert scale to represent instructor perspectives of the given statements: strongly disagree, disagree, neutral, agree and strongly agree. We also include additional space for instructors to record their opinions in the other (please specify) column.

**Question 3: Why do you use multiple choice questions as an instrument for summative assessment?**
Other (please specify):

**Question 4**

In Question 4 instructors are required to express their levels of confidence towards multiple choice questions in summative assessments. We provide five scales for instructors to express
their confidence that multiple choice questions would suit the criteria listed. The scales are: no confidence at all, not confident, neutral, confident and completely confident.

**Question 4:** How confident are you that multiple choice questions in the final exam will do any of the following?

<table>
<thead>
<tr>
<th></th>
<th>No confidence at all</th>
<th>Not confident</th>
<th>Neutral</th>
<th>Confident</th>
<th>Completely confident</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. Test student understanding of programming concepts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ii. Encourage students to think carefully to select the best answer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iii. Encourage students to guess the answer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other (please specify):

**Question 5**

Following on from questions 3 and 4, and concerned with multiple choice questions in summative assessments, we ask the instructors an open ended question about how they devise the distractors for multiple choice questions.

**Question 5: How do you devise the distractors?**
CHAPTER 6. INSTRUCTOR PERSPECTIVES OF MULTIPLE CHOICE QUESTIONS

Question 6

In questions 3 to 5 we explored issues related to multiple choice questions. We further investigated instructor views about devising questions for summative assessments in questions 6 and 7. Question 6 asks the instructor what factors they consider when they devise the tasks or questions.

Question 6: What factors do you consider when you devise exam questions?

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. The question needs to cover a particular programming topic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ii. The question needs to have a certain level of difficulty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other (please specify):

Question 7

The final question in Section A is an open-ended question that seeks instructor views of summative assessments as measures of the student ability to read and write programs.

Question 7: Do you think summative assessment is a valid measure of student ability to program?

6.2.2 Section B

In Chapter 5 we investigated instructors levels of complexity and novice levels of difficulty (refer Section 5.4.2 and Section 5.4.3). In Section 5.4.6 we reported that we found correlation between the variables tested. Therefore, in this chapter, we investigate deeper the level of difficulty, to continue with the theme of the levels of difficulty tested in the questions in summative assessment, from the instructor perspectives according to few criteria as discussed in the next paragraphs.

In Section B of this survey, we sought instructor evaluations of skills or knowledge tested. According to Shneiderman and Mayer, part of the knowledge stored in the memory, about programming concepts and techniques are syntax and semantic knowledge [Shneiderman and Mayer, 1979]. Semantic knowledge includes general programming concepts, regardless
of the language taught. Syntax knowledge is the precise, detailed and arbitrary compared to semantic knowledge. We are also interested to know the level of difficulty and problem solving skills tested in the programming questions. Hence we attempt to study the syntax knowledge, semantic knowledge and problem solving and level of difficulty. For each of the criteria (syntax knowledge, semantic knowledge, problem solving and level of difficulty), instructors need to choose between low, medium or high based on their evaluation to the question.

Although the survey is open for the levels of the programming courses as specified in Section 6.2.1, we highlighted in the survey that questions 8, 9, 10 and 11 are targeted to be tested on the level of novice programmers in introductory courses and instructors’ evaluations should reflect to this condition.

The instructors were asked to evaluate three multiple choice questions (Questions 8, 9 and 10) plus one short answer question (Question 11).

Question 8 was drawn from the course Programming 1 and the other three (Questions 9, 10 and 11) from the BRACElet set (refer Table 3.7 for more information about the BRACElet research group), including: one multiple choice question, one Parson’s Puzzle question [Parsons and Haden, 2006] and one SOLO thinking-out-loud question [Lister et al., 2006].

Questions 9, 10 and 11 in the survey have been previously tested in previous semester final examinations. Furthermore, Question 9 was a benchmark for the student achievements. None of the four questions requires code writing skills but, instead, each tests code-reading and code-comprehension of selected concepts. We discuss these questions further in the following sections. The questions are presented under the headings of Question 8, Question 9, Question 10 and Question 11.

**Question 8**

The sample program in Question 8 is delivered during week one of our course, Programming 1. The students encounter it as part of their formative assessment in the course. We include the question as a summative assessment question to find out how it might be evaluated by the instructors.

```java
public class Reftest {
    public static void main (String argvs[ ]) {
        String s;
    }
```
Question 8: What is the output of the program above?
A. s is now referring to Apple
B. s is now referring to Orange
C. s is now referring to Banana
D. s is now referring to Apple Orange Banana

The objective of Question 8 is to test three concepts. Firstly, it tests the understanding of the term variable. Students must show whether a variable can hold one or more values, simultaneously, or values belonging to several data types, and understand the general concept of a variable. The second aim is to test the understanding of a program in execution by tracing the instructions sequentially. Finally, the question tests the concept of instantiation and its impact on the creation of a new instance of a data type (in this case, a class). The answer for Question 8 is the multiple choice option C, “s is now referring to Banana”.

Question 9

Question 9 is from the group of multiple choice questions from the BRACElet set of questions [Lister et al., 2004; Parsons and Haden, 2006]. This question was found to be the most difficult one for the Leeds Group’s novices, with 62% correct as a benchmark [Lister et al., 2004]. We used it in our summative assessment (Semester 2, 2008) and, in our random sampling of 55 students, found that 62.27% of the students chose a correct response for this question. The question tests logic and problem solving skills instead of syntax knowledge.

Consider the following code fragment:

```java
int[ ] x1 = {0, 1, 2, 3};
int[ ] x2 = {1, 2, 2, 3};
int i1 = 0;
int i2 = 0;
int count = 0;

while ( (i1 < x1.length) && (i2 < x2.length))

112
CHAPTER 6. INSTRUCTOR PERSPECTIVES OF MULTIPLE CHOICE QUESTIONS

{  
    if ( x1[i1] == x2[i2] )  
    {  
        ++count;  
        ++i2;  
    }  
    else if (x1[i1] < x2[i2])  
    {  
        ++i1;  
    }  
    else  
    {  
        // x1[i1] > x2[i2]  
        ++i2;  
    }  
}

Question 9: After this code is executed, “count” contains:

a) 0  
b) 1  
c) 2  
d) 3  
e) 4  

The answer for this question is option E, “After this code is executed, count contains the value 4”. This question tests understanding of simple array processing via loops and conditional statements. In short, the question tests sequence, selection, iteration and array concepts.

Question 10

Question 10 is a Parson’s Puzzle question [Parsons and Haden, 2006]. A Parson’s Puzzle question may be developed as a multiple choice question. A Parson’s Puzzle question presents a jumbled set of lines of code and is based on the premise that learning to program is similar to learning a new language. Such questions test whether the novice can organise the various lines of code into a coherent program. This question involves three major components of programming: sequence, selection and iteration.

Question 10: The lines of code provided below are jumbled. When the lines are correctly ordered, the code calculates the average of the numbers stored in the array $x$. 

113
CHAPTER 6. INSTRUCTOR PERSPECTIVES OF MULTIPLE CHOICE QUESTIONS

When the code above is correctly ordered, in what order would some of the lines occur?

a) sum = 0 before if (Count > 0) { before sum = sum + x[Count];

b) sum = 0 before count++ before if (Count > 0) {

c) sum = sum + x[Count]; before sum = 0 before count++

d) sum = sum + x[Count]; before count++ before sum = 0

A study on the evaluation of Parson’s Puzzle questions reported that if students were given a choice between rearranging given code statements and writing their own code, they preferred the latter [Denny et al., 2008a], and often this was expressed as preferring to “solve things my own way”. In another response it was pointed out that in comparison to other exam question styles it requires a different technique: “in multiple choice examination questions, I would come up with an answer before choosing an option”, but, the student is not able to do this with a Parson’s Puzzle question. A few students did recognise the benefit of a Parson’s Puzzle question in an examination situation as it tests problem solving to a greater extent with one student suggesting that it also provides hints about the solution.

**Question 11**

We further our research to study a SOLO thinking-out-loud question [Lister et al., 2006]. This is not a multiple choice question as novices are not provided options for the answer. A SOLO (Structure of the Observed Learning Outcome) thinking-out-loud question is loosely based on the premise that learning to program may be likened to learning to communicate, where one step might be to read code but another step might be to explain and verbalise what the code performs.

Question 11: Suggest a name for method10 below that reflects its purpose:
public float method10(int[] aiNumbers) {
    int iSum = 0;
    for (int iLoop = 0; iLoop < aiNumbers.length; iLoop++) {
        iSum += aiNumbers[iLoop];
    }
    return iSum / aiNumbers.length;
}

In this question, novices are required to provide a short answer describing what the program does. This question presents a program that calculates the average numbers which are read through a loop.

In the next section we explain the criteria for the evaluation of the questions explained above.

6.3 Criteria for Instructor Levels of Complexity

In this section we introduce four criteria to establish the instructor levels of complexity proposed in the previous chapter. Instructors could use these criteria as a guide to evaluate the questions appearing in Section B of the survey. The criteria are syntax knowledge, semantic knowledge, problem solving skills and level of difficulty.

This is how we defined the criteria for the instructor levels of complexity in the survey:

- **Syntax knowledge** is the knowledge relating to the programming language. This includes the grammar and the flow of the language.

- **Semantic knowledge** is the knowledge relating to the interpretation of language constructs. This includes concepts, principles, rules and problem solving strategies.

- **Problem solving skills** is a measure of the problem solving skills needed to solve the question for an average student.

- **Level of difficulty** may be measured by the complexity of the question and the number of skills or concepts tested.

We have discussed the design of our survey and in the next section we report our findings from the survey.
6.4 Instructor Survey Responses

We received 66 responses to the survey. As previously mentioned, the responses were anonymous and the data was collected from all over the world, including from members of the SIGCSE, PPIG, ITICSE research groups (refer to Table 3.7 for more information about these research groups) and RMIT University teaching staff. In the following sections we present summaries of the responses received in line with the way the survey was structured (refer Figure 6.1):

Instructor Backgrounds, Instructor General Perspectives of Summative Assessment, Instructor Perspectives on Multiple Choice Questions, and Instructor Evaluations of Particular Questions. We provided the details of the survey in Section 4.11. The raw data collected from the programming instructors are included in Appendix D.2.

In our analysis of the survey in chapter, we refer to novices as students because the instructors may able to relate their teaching role to the students. Furthermore, in the open ended questions, instructors are more familiar to use the terms students compared to novices.

6.4.1 Instructor Background and Experiences

First we look at the instructor experiences in relation to the programming courses. The survey revealed a range of one to 74 semesters of teaching experience among instructors with an average teaching experience of 20.44 semesters.

We use the semester to measure experience instead of a year as it is more accurate to do so. One year may variably contain two or three teaching semesters, including potentially summer, autumn, spring or winter semesters. Therefore it is inefficient to measure the instructor experiences based on yearly as in a year there can be a few number of semesters. Plus, at the end of each semester an instructor may modify their teaching and devise the assessments differently. The summative assessments or final examinations are produced for each semester. Thus the instructor experiences are best measured by the semesters.

We perform crosstabulation and present the information about instructor experiences of teaching and preparing exam questions in Table 6.1. Both types of such experiences are categorised into three groups based on the experience over a number of semesters.

Referring to Table 6.1 we group the instructors into three groups based on their semesters of experience (0 to 19 semesters, 20 to 39 semesters and, 40 and onwards). Horizontally, the groups are labelled as $T_{i\rightarrow j}$, indicating teaching experience in the range of $i$ to $j$ semesters. For example, $T_{20\rightarrow 39}$ indicates the instructors who have been teaching programming between
CHAPTER 6. INSTRUCTOR PERSPECTIVES OF MULTIPLE CHOICE QUESTIONS

Table 6.1: Descriptions of Instructors Grouped by Semesters of Experience in Teaching (Teaching) and in Devising Examination Questions (Devising Question)

<table>
<thead>
<tr>
<th>Count</th>
<th>$D_{0-19}$</th>
<th>$D_{20-39}$</th>
<th>$D_{40+}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teaching $T_{0-19}$</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>% within Devising Questions</td>
<td>91.9%</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>% of Total</td>
<td>51.5%</td>
<td>0</td>
<td>0</td>
<td>51.5%</td>
</tr>
<tr>
<td>$T_{20-39}$</td>
<td>3</td>
<td>21</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>% within Devising Questions</td>
<td>8.1%</td>
<td>95.5%</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>% of Total</td>
<td>4.6%</td>
<td>31.8%</td>
<td>0</td>
<td>36.4%</td>
</tr>
<tr>
<td>$T_{40+}$</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>% within Devising Questions</td>
<td>0</td>
<td>4.5%</td>
<td>100%</td>
<td>-</td>
</tr>
<tr>
<td>% of Total</td>
<td>0</td>
<td>1.5%</td>
<td>10.6%</td>
<td>12.1%</td>
</tr>
<tr>
<td>Total</td>
<td>37</td>
<td>22</td>
<td>7</td>
<td>66</td>
</tr>
<tr>
<td>% within Devising Questions</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>% of Total</td>
<td>56.1</td>
<td>33.3</td>
<td>10.6</td>
<td>100</td>
</tr>
</tbody>
</table>

20 and 39 semesters and $T_{40+}$ indicates the instructor who have been teaching programming for more than 40 semesters. Vertically, $D_{i-j}$ symbolises experience in devising examination questions. As an example, $D_{20-39}$ indicates the instructors who have been devising programming questions between 20 and 39 semesters.

More than half of the instructors who responded to the survey have been teaching and devising questions for between none and 19 semesters (highlighted in green, refer Table 6.1).

All the instructors who responded to the survey had the experience in teaching programming courses, however, based on the data collected we found that there were two instructors in $T_{0-19}$ group who did not have the experience in devising examination questions (zero semester of experience in devising examination questions). They may be new programming instructors because they have very few semesters of experience in teaching programming courses. We would expect the number of semesters in devising questions as lesser than the number of semesters in teaching as it will require the instructor to teach the course for some time in order to gain the experience on what and how to test the students in their assessment.

There are 4.6% of the responses with the experience in teaching between 20 and 39 semesters, and none to 19 semesters experience in devising examination questions. Table 6.1 shows that only four instructors who responded to the survey have less experience in devising
questions than they have in teaching (highlighted in yellow). Typically one would expect instructors to have close to equal numbers of semesters of experience in teaching and devising questions, a phenomenon evident by the presence of the largest numbers being on the diagonal of the matrix presented in the table (highlighted in blue, refer Table 6.1).

Overall 56.1% of the responses have zero to 19 semesters experience in devising questions, 33.3% of the responses have 20 to 39 semesters experience in devising questions and 10.6% of the responses have 40 semesters onwards experience in devising questions.

There are 51.5% of the responses have zero to 19 semesters experience of teaching programming courses and devising programming questions. There are 36.4% of the responses have 20 to 39 semesters experience in devising questions and 12.1% of the responses have 40 semesters onwards experience in devising questions.

As the survey was distributed to instructors of programming courses, in its generic sense, we are interested to establish the distribution of instructors teaching across individual (programming) courses.

Instructors were asked to select one or more courses that best represented their teaching experience. As instructors may select more than one option, overlapping may occur. Our main concern is to know their experience teaching in multiple programming courses, therefore we allow them to include all their experience teaching the different levels of programming courses. From the responses to Question 2, we find that 47% of instructors have taught Introduction to Programming, 94% of instructors have taught Programming 1, while 85% have taught Programming 2, and 48% have taught Programming 3.

In this section we have studied the instructor backgrounds and in the next section we present their views of summative assessment.

### 6.4.2 Instructor Views of Summative Assessment

In this section we present the instructor views about the reliability of summative assessment as a valid measure of student ability to program. We also discuss the factors they consider when devising questions in Table 6.2, Question 7 of the survey.

The data collected from Question 7 of the survey are presented in Table 6.2, the elicited views about summative assessment as a valid measure of a student’s ability to program. Table 6.2 also presents Question 6 of the survey regarding factors the instructors considered when devising exam questions.

Please note that *no opinion* responses in Table 6.2, Table 6.4, Table 6.6 and Table 6.8
mean that no response was given to that particular question. This was not an option provided in the survey, but included in our results because several instructors did not respond to some of the questions.

**Table 6.2: Instructor Views of Summative Assessment**

<table>
<thead>
<tr>
<th>Question</th>
<th>Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Opinion</td>
</tr>
<tr>
<td>7. Do you think summative assessment is a valid measure of a student’s ability to program?</td>
<td>3.03</td>
</tr>
<tr>
<td>6. Factors to be considered when devising questions:</td>
<td></td>
</tr>
<tr>
<td>a) The question needs to cover a particular programming topic</td>
<td>1.52</td>
</tr>
<tr>
<td>b) The question needs to have a certain level of difficulty</td>
<td>1.52</td>
</tr>
</tbody>
</table>

We report that 48.49% of the instructors agreed or strongly agreed while 21.21% disagreed or strongly disagreed that summative assessment is a valid measure of a student’s ability to program. 27.27% were neutral, while 3% left the question blank, which we record as no opinion. Hence, we can conclude that the almost half of the responses agreed that summative assessment can provide a valid measure of student ability to program.

We further investigate the factors instructors consider when they devise the questions (Table 6.2, Question 6). More than 90% of the instructors agreed or strongly agreed that the exam questions need to cover particular programming topics while 83.3% felt they had to have a certain level of difficulty. There were also some other responses added by the instructors about how they devise questions. We categorise these responses and present them in Table 6.3. They cover issues such as the amount of time a student has to complete the exam, the focus of the assessment, how much coverage has been given to a particular topic, question re-use and validity of the questions.

There were two responses from the instructors that mentioned that they rely on free
Table 6.3: Other Responses to the Factors Instructors Consider When They Devise Questions (Question 6)

<table>
<thead>
<tr>
<th>Theme</th>
<th>Quotations from Instructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>“The student must be able to determine how much time to commit and the time must be really short.”</td>
</tr>
<tr>
<td></td>
<td>“It can be solved in a limited time frame, without resource to a computer.”</td>
</tr>
<tr>
<td>Focus</td>
<td>“The question has to be presented in a way in which the concepts behind the question are the focus, rather than understanding the context and/or detail of the question itself.”</td>
</tr>
<tr>
<td>Coverage</td>
<td>“I try to include enough questions to cover all the important concepts.”</td>
</tr>
<tr>
<td></td>
<td>“Should complement other questions, even test again to ensure the student did not just get lucky on one question.”</td>
</tr>
<tr>
<td></td>
<td>“I also ask questions directly from the textbook and lectures.”</td>
</tr>
<tr>
<td></td>
<td>“I try to ensure the student learned certain concepts from class and from outside projects.”</td>
</tr>
<tr>
<td></td>
<td>“The question shouldn’t be hard - just the material it covers.”</td>
</tr>
<tr>
<td></td>
<td>“How much foundational knowledge is necessary and how much application rather than cognition?”</td>
</tr>
<tr>
<td></td>
<td>“The material has been covered adequately in the course.”</td>
</tr>
<tr>
<td>Revision</td>
<td>“Have I used this question before? If so, need to alter it for a second use.”</td>
</tr>
<tr>
<td>Validity</td>
<td>“I think free response questions have high validity; I have much less confidence in multiple choice questions.”</td>
</tr>
<tr>
<td></td>
<td>“You need to include essay questions because Computer Science students need to know how to write.”</td>
</tr>
</tbody>
</table>

response questions such as essays and short answer questions to test students. The rationale behind it is that in an assessment of programming, the students need to be tested on their ability to write a program.

Therefore the test instruments do play an important role in order to test the students’ abilities in programming. In the next section we focus on multiple choice questions as an instrument in summative assessment.
6.4.3 Instructor Views of Multiple Choice Questions

In this section we look at the reasons instructors use multiple choice questions. Table 6.4 summarises the results of Question 3, Section A of our survey.

Table 6.4: Question 3: Why Do You Use Multiple Choice Questions for Summative Assessment?

<table>
<thead>
<tr>
<th>Description</th>
<th>Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Opinion</td>
</tr>
<tr>
<td>a) Easy to mark</td>
<td>16.67</td>
</tr>
<tr>
<td>b) Precise answer provided</td>
<td>18.18</td>
</tr>
<tr>
<td>c) To give some hint of the fragment’s code to the next section of the assessment</td>
<td>18.18</td>
</tr>
<tr>
<td>d) To test students on a low level of understanding</td>
<td>18.18</td>
</tr>
</tbody>
</table>

Almost 70% of the instructors agreed or strongly agreed that multiple choice questions are easy to mark, provide precise answers and 50% of the instructors agreed or strongly agreed that multiple choice questions test students on low levels of understanding. Only 25.76% of the instructors agreed or strongly agreed, 28.79% were neutral and, 27.27% disagreed or strongly disagreed that multiple choice questions and its distractors may give hints to novices in relation to answering the code fragments questions appearing in subsequent sections of the same assessment.

Complementing the quantitative summaries, Table 6.5 presents the qualitative responses from instructors. The instructors use multiple choice questions to test the students’ understanding of fundamental programming concepts. An instructor emphasised that multiple choice questions may gain the confidence of the weak students to answer the questions. Other instructors mentioned that multiple choice questions can be used to test the depth of knowledge of the students. Therefore the instructors think that multiple choice questions can test multiple levels of knowledge and ability, not only restricted to test on the students’ understanding of programming concepts. One of the instructors stated that using multiple choice questions, students are encouraged to analyse the options provided in order to choose the best answer. According to Bloom’s Taxonomy analysis is a high level of cognitive process,
which is in rank number three after create and evaluate (refer Figure 3.2). Therefore, the process of answering multiple choice questions may require a high level of cognitive process.

Table 6.5: Reasons Instructors Use Multiple Choice Questions

<table>
<thead>
<tr>
<th>Theme</th>
<th>Quotations from Instructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidence</td>
<td>“To give weaker students confidence to answer questions.”</td>
</tr>
<tr>
<td>Understand</td>
<td>“To get an idea of the breadth of students’ understanding.”</td>
</tr>
<tr>
<td></td>
<td>“Good to reveal whether they really understand a particular concept.”</td>
</tr>
<tr>
<td>Easy question</td>
<td>“Provide a series of easier questions where there is a high likelihood that students who have studied should be able to answer correctly.”</td>
</tr>
<tr>
<td></td>
<td>“To test understanding of fundamental terms/phrases used in programming.”</td>
</tr>
<tr>
<td>Feedback</td>
<td>“Immediate feedback for the students when preparing for the assessment if the appropriate tools are in place.”</td>
</tr>
<tr>
<td></td>
<td>“Shorter feedback time.”</td>
</tr>
<tr>
<td>Level of knowledge</td>
<td>“Gives common misperceptions as possible answers to help define exact level of knowledge.”</td>
</tr>
<tr>
<td></td>
<td>“Multiple choices can also be used to test the depth of knowledge.”</td>
</tr>
<tr>
<td></td>
<td>“To test situations where students would normally have other cues not provided on the test (like IDE contents).”</td>
</tr>
<tr>
<td>To gain analysis skill</td>
<td>“Encourage students to analyse different responses noting differences and choosing the best alternative.”</td>
</tr>
<tr>
<td></td>
<td>“I think it provides many options for students and they can decipher which is the right one - gets them thinking.”</td>
</tr>
<tr>
<td>Student centred</td>
<td>“To keep students happy.”</td>
</tr>
<tr>
<td></td>
<td>“To constrain the students’ creativity.”</td>
</tr>
</tbody>
</table>

Next, we summarise instructor confidence in multiple choice questions (refer Table 6.6). Over 46% of the instructors felt confident or completely confident that multiple choice questions test students’ understanding of programming concepts and encourage students to think carefully before selecting the best answer. However, there were mixed views about the statement “encourage the students to guess the answer”. Multiple choice questions may be popular with the students because they are able to guess an answer if they do not know it. Therefore, we would like to know the instructor views about this issue. We received 34.85% who chose
TABLE 6.6: Question 4: How Confident Are You That Multiple Choice Questions In The Final Exam Will...

<table>
<thead>
<tr>
<th>Description</th>
<th>Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Opinion</td>
</tr>
<tr>
<td>a) Test the student’s understanding of programming concepts</td>
<td>9.09</td>
</tr>
<tr>
<td>b) Encourage the students to think carefully to select the best answer</td>
<td>10.61</td>
</tr>
<tr>
<td>c) Encourage the students to guess the answer</td>
<td>9.09</td>
</tr>
</tbody>
</table>

The neutral column, yet collectively more than 40% were confident or completely confident that the statement was true.

The responses to Question 4b and 4c were inconclusive, because more than 40% of instructors were confident or completely confident that multiple choice questions encouraged students to think carefully as well as that multiple choice questions encouraged students to guess the answer.
We gather the responses to the open ended question about instructor confidence in multiple choice questions and present it as below.

- “Good multiple choice questions can do these things”
- “Test specific theory”
- “Lead up to final assessment as I use these during course too”
- “Although I use them, I regard multiple choice questions as ”free points”, not really diagnostic, but keeps the weaker students on track”
- “Some think carefully; some guess”
- “It depends on how you weight wrong answers”
- “Part of the use of multiple choice questions is to prepare students for later courses where these will be even more common”

We are curious about one of the response, that mentioned “good multiple choice questions can do these things” It is a qualified response and suggests that multiple choice questions are valid assessment instruments, but quality must come first in the matter of creating and evaluating the question. It prompts us to ask the further questions:

- How do instructors define and create good multiple choice questions?
- How do instructors create the distractors to test the student’s abilities?

Distractors play an important role in multiple choice questions. A good distractor may capture the students’ misconceptions about programming concepts. We seek to understand how instructors create the distractors for multiple choice questions. Table 6.7 summarises responses from instructors as how they devise distractors.

The majority of instructors responses mentioned that they devise distractors based on common mistakes made by students, and devise answer options that are close to the correct answer in order to test whether the student really understands the concept. Those students who have misconceptions may be fooled by the distractors. Despite the positive responses to the use of multiple choice questions, we also received the following statements from instructors who do not use multiple choice questions in the summative assessment:

- “I do not devise distractors, free response questions seem more appropriate to me.”
<table>
<thead>
<tr>
<th>Theme</th>
<th>Quotations from Instructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Mistakes</td>
<td>“Questions the students have asked in class.”</td>
</tr>
<tr>
<td></td>
<td>“All possible common errors that students may make when examining the codes.”</td>
</tr>
<tr>
<td></td>
<td>“Observed errors or typical mistakes that occur when students have a concept confused.”</td>
</tr>
<tr>
<td></td>
<td>“Confusing terms or concepts (e.g. polymorphism Vs. encapsulation Vs. inheritance).”</td>
</tr>
<tr>
<td></td>
<td>“Previous student mistakes. If the students don’t understand, the mistakes are often repeated.”</td>
</tr>
<tr>
<td></td>
<td>“To make a certain error and follow it.”</td>
</tr>
<tr>
<td></td>
<td>“By choosing answers that are likely to be picked by students with common misunderstandings.”</td>
</tr>
<tr>
<td>Close to Answer</td>
<td>“I try to make them similar to the real answer, and at least sound plausible to a student who is not confident in the correct answer so that a guess is harder and thought is required to select correct answer.”</td>
</tr>
<tr>
<td></td>
<td>“I tend to do it in pairs - the correct answer and one that is close to it but is erroneous due to a commonly made assumption that is incorrect, and another pair which also look reasonable but have a fundamental flaw which allows them to be eliminated with a basic understanding of the concept behind the question.”</td>
</tr>
<tr>
<td>Random</td>
<td>“If I’m really stuck for that last distractor, it might be some random correct “looking” answer, or a comedy option.”</td>
</tr>
<tr>
<td></td>
<td>“I use familiar-sounding terminology, terminology from other areas of the course, and sometimes humour.”</td>
</tr>
</tbody>
</table>

- “I don’t use multiple choice questions.”
- “I have NEVER used multiple choice questions in an exam!”
- “I don’t put any multiple choice questions on an exam. I do use them on weekly quizzes on the first few quizzes. I usually use multiple choice questions for definition of terms problems, in which case I use other terms covered during the same time period. I also use multiple choice questions for “step-through” problems. The distractors in that case would represent the various wrong ways that I’ve seen students step through the code.”
- “I feel that multiple choices is a completely inappropriate tool for judging deep understanding and comprehension of programming concepts.”
An instructor mentioned that multiple choice questions are only used in quizzes, specifically to test on the definitions or terminologies and step through the code. Quizzes could be used as a formative assessment, and through the quizzes, multiple choice questions may benefit the students in learning fundamental programming concepts. This instructor’s opinion convinced us to use multiple choice questions in our proposed learning tool, as it is a formative assessment and the proposed module covers fundamental of programming concepts.

In the next section, we study the instructor responses to a few sample of examination questions.

6.4.4 Instructor Evaluations of Multiple Choice Questions

In this section we look at instructor responses to the four questions, namely questions 8, 9, 10 and 11. The questions have been explained in detail in Section 6.2 and in this section we discuss the responses from the instructors based on the syntax knowledge, semantic knowledge, problem solving skills and level of difficulty tested in the questions. In Section 6.2.2 we stated that in the survey we provide three scales (low, medium and high) for the instructors to select based on their evaluation to the questions.

Table 6.8 outlines the details of the distribution of responses for these questions. Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5 shows the distribution of responses for each of questions 8, 9, 10 and 11.

![Figure 6.2: Instructor Responses to Question 8 Based on Syntax Knowledge, Semantic Knowledge, Problem Solving Skill and Level of Difficulty](image1)

Figure 6.2: Instructor Responses to Question 8 Based on Syntax Knowledge, Semantic Knowledge, Problem Solving Skill and Level of Difficulty
Table 6.8: Frequency of Data for Levels of Skills and Knowledge Tested in Questions 8, 9, 10 and 11

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Criteria</th>
<th>No Opinion</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Syntax Knowledge</td>
<td>1.52</td>
<td>69.70</td>
<td>19.70</td>
<td>9.09</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>1.52</td>
<td>37.88</td>
<td>42.42</td>
<td>18.18</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>1.52</td>
<td>78.79</td>
<td>18.18</td>
<td>1.52</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>1.52</td>
<td>78.79</td>
<td>16.67</td>
<td>3.03</td>
</tr>
<tr>
<td>9</td>
<td>Syntax Knowledge</td>
<td>3.03</td>
<td>15.15</td>
<td>56.06</td>
<td>25.76</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>4.55</td>
<td>4.55</td>
<td>56.06</td>
<td>34.58</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>3.03</td>
<td>22.73</td>
<td>43.94</td>
<td>30.30</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>4.55</td>
<td>3.03</td>
<td>54.55</td>
<td>37.88</td>
</tr>
<tr>
<td>10</td>
<td>Syntax Knowledge</td>
<td>4.55</td>
<td>18.18</td>
<td>62.12</td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>6.06</td>
<td>7.58</td>
<td>46.97</td>
<td>39.39</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>4.55</td>
<td>13.64</td>
<td>37.88</td>
<td>43.94</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>4.55</td>
<td>4.55</td>
<td>53.03</td>
<td>37.88</td>
</tr>
<tr>
<td>11</td>
<td>Syntax Knowledge</td>
<td>4.55</td>
<td>39.39</td>
<td>51.52</td>
<td>4.55</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>4.55</td>
<td>15.15</td>
<td>60.61</td>
<td>19.70</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>4.55</td>
<td>37.88</td>
<td>48.48</td>
<td>9.09</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>7.58</td>
<td>42.42</td>
<td>48.48</td>
<td>1.52</td>
</tr>
</tbody>
</table>

For Question 8 almost 70% of the responses indicated that the question rated as requiring a low level of syntax knowledge (refer Table 6.8). 37.88% of the responses rated the question as a low level of semantic knowledge and 42.42% of the responses rated the questions as a medium level of semantic knowledge. The majority (78.79%) of the responses rated the question as low level of difficulty and problem solving skills. As the question was selected from the first week of our course module, we anticipated it might require low levels of proficiencies in all of the categories, although a medium level of semantic knowledge here is understandable, because it is where any difficulty in this question may exist.

As explained in Section 6.2.1 Question 9 is the most difficult in the BRACElet set of questions. Question 9 tests on student ability to solve problems and application of their knowledge of sequence, iteration and array concepts. The part that emphasise to test novices’ knowledge of the syntax is “//x1[11] > x2[12]” whereby if novices know that the line is commented out, then the line of the program is not executed.

For Question 9 the more than 50% of the responses identified the question as requiring a
Figure 6.3: Instructor Responses to Question 9 Based on Syntax, Semantic Knowledge, Problem Solving Skill and Level of Difficulty

Figure 6.4: Instructor Responses to Question 10 Based on Syntax, Semantic Knowledge, Problem Solving Skill and Level of Difficulty

medium level of syntax knowledge and a medium level of semantic knowledge (refer Table 6.8). 43.94% of the responses indicated the question as a medium level of problem solving skills, 22.73% of the responses indicated the question as a low level of problem solving skills and 30.30% of the responses indicated the question as a medium level of problem solving skills. We
received variability of instructors responses to this question. More than half of the responses rated the question as a medium level of difficulty and 37.88% of the responses rated the question as a high level of difficulty.

For Question 10, most of the instructors who responded to the survey identified the question as requiring a medium level of syntax knowledge (refer Table 6.8). This question is a Parson Puzzle question, it requires students to rearrange the lines of code to become a working program. Hence, it does test on the students’ ability to understand the syntax and to arrange them accordingly. 46.97% of the responses rated the question as a medium level of semantic knowledge and almost 40% of the responses rated as the high level of semantic knowledge. Based on the instructor views, 37.88% of the responses rated the question as a medium level of problem solving skills and 43.94% of responses rated the question as a high level of problem solving skills. More than half of the responses received rated Question 10 as a medium level of difficulty and 37.88% of the responses rated the question as a high level of difficulty.

This may be because different arrangements of the given lines of code potentially yield different results. Moreover, the answer and the distractors are very similar and thus, strong problem solving skills are required in order to answer this question correctly. The syntax knowledge are provided, so the question does not rate as highly in terms of these measures. Only 15.15% of the responses rated this question as a high level of syntax knowledge.
For Question 11, 51.52% of the responses indicated that the question requires a medium level of syntax knowledge, almost 40% indicated that the question requires a low level of syntax knowledge and only 4.55% indicated this question is at high level of syntax knowledge (refer Table 6.8). This question requires students to read the program and identify the purpose of the program. As it does not require the ability to write the program, that may be why the majority of the responses chose low and medium levels of syntax knowledge. More than 60% of the responses rate this question as a medium level of semantic knowledge. 37.88% of the responses rated the question as a low level of problem solving skills, 48.48% of the responses rated the question as a medium level of problem solving skills, and only 9.09% of the responses rate the question as a high level of problem solving skills. The same pattern is observed in the level of difficulty, whereby, majority of the responses rated the question as low and medium level of difficulty.

Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5 present the instructor responses to questions 8, 9, 10 and 11. Based on the pattern of the responses from Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5, we can see that majority of the responses for Question 8 is at the low levels for syntax knowledge, problem solving skills and level of difficulty, and majority of the responses chose low and medium levels of semantic knowledge (refer Figure 6.2).

For Question 9 majority of the responses chose a medium level for the four criteria (refer Figure 6.3). Some rated high and a few rated low for the four criteria.

For Question 10 majority of the responses rated the question as a medium level of syntax knowledge, and medium and high levels of semantic knowledge, problem solving skills and the level of difficulty.

The SOLO thinking-out-loud question (Question 11) is in the medium levels of the four criteria but there are also a number of responses for the low levels of the syntax knowledge, problem solving skills and the level of difficulty.

We have presented the data collected based on the frequency. We received mixed responses and were not able to conclude the category that instructors chose for each of the criteria listed in the instructor levels of complexity. Therefore we present the mean for each of the questions based on the criteria of the instructor levels of complexity.

There was no response given to the questions by few of the instructors. Again, we emphasise that we added the no opinion category to the three existing scales low, medium and high in Table 6.9 to represent these group of instructors. The weightage for no opinion is zero, low is one, medium is two and high is three. We present the mean for each question (Table 6.9) and provide the range for each category (Table 6.10). We round the numbers
calculated for the mean to the nearest two decimal places.

Table 6.9: Mean for Levels of Skills and Knowledge Tested in Questions 8, 9, 10 and 11

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Criteria</th>
<th>Mean</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Syntax Knowledge</td>
<td>1.36</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>1.77</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>1.20</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>1.21</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td><strong>Overall Mean</strong></td>
<td>1.39</td>
<td>Low</td>
</tr>
<tr>
<td>9</td>
<td>Syntax Knowledge</td>
<td>2.05</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>2.21</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>2.02</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>2.26</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td><strong>Overall Mean</strong></td>
<td>2.14</td>
<td>Medium</td>
</tr>
<tr>
<td>10</td>
<td>Syntax Knowledge</td>
<td>1.88</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>2.20</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>2.21</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>2.24</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td><strong>Overall Mean</strong></td>
<td>2.13</td>
<td>Medium</td>
</tr>
<tr>
<td>11</td>
<td>Syntax Knowledge</td>
<td>1.56</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Semantic Knowledge</td>
<td>1.95</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Problem Solving Skills</td>
<td>1.62</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Level of Difficulty</td>
<td>1.44</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td><strong>Overall Mean</strong></td>
<td>1.64</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 6.10: Range of the Means for Each Category

<table>
<thead>
<tr>
<th>Range</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 0.49</td>
<td>No opinion</td>
</tr>
<tr>
<td>0.50 to 1.49</td>
<td>Low</td>
</tr>
<tr>
<td>1.50 to 2.49</td>
<td>Medium</td>
</tr>
<tr>
<td>2.50 to 3.00</td>
<td>High</td>
</tr>
</tbody>
</table>

The overall mean for Question 8 is 1.39 which is classified as *low* in *instructor levels of complexity*. This question is from the first week modules of an introductory programming course. Based on the calculation of the means, this question tests on the low level of *syntax knowledge*, *problem solving skills*, *level of difficulty*, and medium in *semantic knowledge*. 

131
The mean for *semantic knowledge* (1.77) is a bit higher than the other criteria tested in the question. The challenging part of this question that contributes to a medium level of semantic knowledge is because novices are required to understand the programming concepts that a variable can hold a value at once and a program executes sequentially.

For Question 9 the overall mean is 2.14 which is classified as *medium* in *instructor levels of complexity*. All the means of the four criteria are *medium* in the *instructor levels of complexity*.

For Question 10 the overall mean is 2.13 which is classified as *medium* in the *instructor levels of complexity*. The mean for the *syntax knowledge* is lower compared to the other three criteria for this question, because this is the Parson’s Puzzle question, whereby novices are requested to rearrange the code to be a meaningful program. Therefore, the instructors may think that this question does not emphasise test by the novice programmers’ syntax knowledge compared to the other criteria assessed.

Question 11 is a short answer question, and all the three questions discussed above are multiple choice questions. Overall, the mean for Question 11 is 1.64 which is classified as *medium* in the *instructor levels of complexity*. Although questions 9, 10 and 11 are in the *medium* level of *instructor levels of complexity*, but based on our calculation for the mean, Question 11 (mean = 1.64) has been rated as lower than questions 9 (mean = 2.14) and 10 (mean = 2.13). Using the *instructor levels of complexity*, we have seen that multiple choice questions can be rated higher than a short answer question in terms of the syntax, semantic knowledge, problem solving skills and level of difficulty.

We presented the data collected in terms of frequencies, bar graphs and means for the *instructor levels of complexity*. In the next section we attempt to study the relationships between the four criteria of *instructor levels of complexity*.

### 6.5 Data Analysis

We test the data collected from instructors using the SPSS statistics package to calculate the correlations, if any, between *syntax knowledge*, *semantic knowledge*, *problem solving skills* and the *level of difficulty*.

SPSS return the values of magnitude for the correlation coefficient ($\rho$) and the p-value which we report in Table 6.11, Table 6.12, Table 6.13 and Table 6.14. If there exist significance of correlation coefficients, the magnitude and direction of the correlation coefficient will be observed based on Table 4.8.
Table 6.11: Correlations Between Variables Measured for Questions 8

<table>
<thead>
<tr>
<th>Variables</th>
<th>$\rho$</th>
<th>$P$-value</th>
<th>Strength of Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Knowledge and Semantic Knowledge</td>
<td>0.469</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Syntax Knowledge and Problem Solving Skills</td>
<td>0.451</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Syntax Knowledge and Level of Difficulty</td>
<td>0.444</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Semantic Knowledge and Problem Solving Skills</td>
<td>0.427</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Semantic Knowledge and Level of Difficulty</td>
<td>0.521</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Problem Solving Skills and Level of Difficulty</td>
<td>0.589</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

Table 6.12: Correlations Between Variables Measured for Questions 9

<table>
<thead>
<tr>
<th>Variables</th>
<th>$\rho$</th>
<th>$P$-value</th>
<th>Strength of Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Knowledge and Semantic Knowledge</td>
<td>0.525</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Syntax Knowledge and Problem Solving Skills</td>
<td>0.360</td>
<td>0.003</td>
<td>Weak</td>
</tr>
<tr>
<td>Syntax Knowledge and Level of Difficulty</td>
<td>0.582</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Semantic Knowledge and Problem Solving Skills</td>
<td>0.381</td>
<td>0.002</td>
<td>Weak</td>
</tr>
<tr>
<td>Semantic Knowledge and Level of Difficulty</td>
<td>0.327</td>
<td>0.000</td>
<td>Weak</td>
</tr>
<tr>
<td>Problem Solving Skills and Level of Difficulty</td>
<td>0.486</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

Table 6.13: Correlations Between Variables Measured for Questions 10

<table>
<thead>
<tr>
<th>Variables</th>
<th>$\rho$</th>
<th>$P$-value</th>
<th>Strength of Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Knowledge and Semantic Knowledge</td>
<td>0.451</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Syntax Knowledge and Problem Solving Skills</td>
<td>0.474</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Syntax Knowledge and Level of Difficulty</td>
<td>0.577</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Semantic Knowledge and Problem Solving Skills</td>
<td>0.705</td>
<td>0.000</td>
<td>Strong</td>
</tr>
<tr>
<td>Semantic Knowledge and Level of Difficulty</td>
<td>0.570</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
<tr>
<td>Problem Solving Skills and Level of Difficulty</td>
<td>0.573</td>
<td>0.000</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

We set the $\alpha$ level to 0.05 (refer Section 5.4.6). All the $p$-values returned were less than 0.05 from Table 6.11, Table 6.12, Table 6.13 and Table 6.14. Therefore there exist significant correlations between all the variables tested as displayed in Table 6.11, Table 6.12, Table 6.13 and Table 6.14.

There exist significant correlations between four variables tested (syntax knowledge, se-
mantic knowledge, problem solving skills and level of difficulty) based on the instructor responses. In other words, when instructors evaluate these questions, the four criteria rated almost equally (if one criterion is low then the other criterion may also be rated as low). Thus, based on result of the correlation coefficient, the students are tested almost equally on the four criteria in questions 8, 9, 10 and 11.

There exist moderate correlations between syntax knowledge and semantic knowledge, syntax knowledge and problem solving skills, syntax knowledge and level of difficulty, semantic knowledge and problem solving skills, semantic knowledge and level of difficulty also problem solving skills and level of difficulty for questions 8 and 11.

For Question 9 there exist moderate correlations between syntax knowledge and semantic knowledge, syntax knowledge and level of difficulty, also problem solving skills and level of difficulty. There exist weak correlations between syntax knowledge and level of difficulty, semantic knowledge and problem solving skills also semantic knowledge and level of difficulty.

For question 10 there exist moderate correlations between syntax knowledge and semantic knowledge, syntax knowledge and problem solving skills, syntax knowledge and level of difficulty, semantic knowledge and level of difficulty also problem solving skills and level of difficulty and there exist a strong correlation between semantic knowledge and problem solving skills. Semantic knowledge and problem solving skills are strongly correlated as rated by the instructors in the Parson’s Puzzle multiple choice question.

We also conduct the correlations between the experience in teaching (refer to Section 6.4.1) and experience in devising exam questions with syntax knowledge, semantic knowledge, problem solving skills and level of difficulty (refer Table 6.15). Note that QNo denotes the question number, and we have explained how we divided the instructors into groups based on their experience in teaching and experience in devising exam questions (refer Table 6.1).
Table 6.15: Correlation between Experience in Teaching and Experience in Devising Exam Questions with the Four Criteria (Syntax Knowledge (SynK), Semantic Knowledge (SemK), Problem Solving Skills (PSS), Level of Difficulty (LD)) for Questions 8, 9, 10 and 11

<table>
<thead>
<tr>
<th>QNo</th>
<th>Variable</th>
<th>SynK</th>
<th>SemK</th>
<th>PSS</th>
<th>LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Experience in Teaching</td>
<td>ρ-value 0.027</td>
<td>-0.004</td>
<td>-0.032</td>
<td>-0.140</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.831</td>
<td>0.975</td>
<td>0.800</td>
<td>0.261</td>
</tr>
<tr>
<td></td>
<td>Experience in Devising Exam Questions</td>
<td>ρ-value 0.043</td>
<td>-0.001</td>
<td>-0.043</td>
<td>-0.150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.734</td>
<td>0.991</td>
<td>0.734</td>
<td>0.228</td>
</tr>
<tr>
<td>9</td>
<td>Experience in Teaching</td>
<td>ρ-value -0.053</td>
<td>-0.068</td>
<td>-0.077</td>
<td>0.029</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.672</td>
<td>0.588</td>
<td>0.577</td>
<td>0.819</td>
</tr>
<tr>
<td></td>
<td>Experience in Devising Exam Questions</td>
<td>ρ-value 0.043</td>
<td>0.018</td>
<td>-0.015</td>
<td>0.154</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.730</td>
<td>0.888</td>
<td>0.907</td>
<td>0.128</td>
</tr>
<tr>
<td>10</td>
<td>Experience in Teaching</td>
<td>ρ-value 0.022</td>
<td>0.008</td>
<td>-0.162</td>
<td>-0.013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.858</td>
<td>0.946</td>
<td>0.194</td>
<td>0.915</td>
</tr>
<tr>
<td></td>
<td>Experience in Devising Exam Questions</td>
<td>ρ-value 0.071</td>
<td>0.084</td>
<td>-0.064</td>
<td>0.046</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.573</td>
<td>0.500</td>
<td>0.609</td>
<td>0.716</td>
</tr>
<tr>
<td>11</td>
<td>Experience in Teaching</td>
<td>ρ-value -0.129</td>
<td>-0.097</td>
<td>-0.221</td>
<td>-0.104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.304</td>
<td>0.440</td>
<td>0.075</td>
<td>0.404</td>
</tr>
<tr>
<td></td>
<td>Experience in Devising Exam Questions</td>
<td>ρ-value -0.089</td>
<td>-0.074</td>
<td>-0.117</td>
<td>-0.047</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-value 0.475</td>
<td>0.555</td>
<td>0.349</td>
<td>0.708</td>
</tr>
</tbody>
</table>

As explained in Section 5.4.6, we set the α level to 0.05. Therefore, if p-value ≤ 0.05, there exists a significant correlation. According to the data collected, all the p-values are more than 0.05, therefore there are no significant correlations between experience in teaching or experience in devising exam questions to any of the criteria syntax knowledge, semantic knowledge, problem solving skills and level of difficulty. In other words, instructor experiences in teaching, lecturing or devising questions have no effect on their evaluation of the multiple choice questions.
6.6 Discussion

A common approach to evaluate novice programming ability is through a combination of formative and summative assessments. Preparation of such assessment is driven by instructor perceptions of student ability to grasp programming concepts. This in turn may yield instructor perspectives of summative assessment that do not necessarily correlate with student expectations or abilities.

In this chapter we present results of our study around instructor perspectives of summative assessment for novice programmers. We have highlighted the instructor views of summative assessment and specifically their views of multiple choice questions as the test instrument for summative assessment for programming courses. Both quantitative and qualitative data have been obtained via survey responses from programming instructors with varying teaching experience to targeted programming questions.

There were 48.49% of responses from the instructors agreed or strongly agreed that summative assessment can provide a valid measure of students’ ability to program. Approximately 21% of the responses disagreed or strongly disagreed that summative assessment is a valid measure to student’s ability to program, and 27.27% chose to be neutral. Hence, that prompted us to ask why there exist a group of instructors that does not consider summative assessment as to measure student’s ability to program and some that would not want to state their stand in regards to summative assessment as appropriate to measuring student ability to program? Also, if they do not think that multiple choice questions are a valid measure of student’s ability to program, what is the more valid assessment for programming? In the survey, few instructors stated that they use essay instead of multiple choice questions to test their student’s ability to program. But, we cannot say for sure that an essay can be a more valid measure for student’s ability to program compared to multiple choice questions. We may investigate this issue in our future research.

In the survey, there were few instructors who did not support the use of multiple choice questions in summative assessment. We are not able to quantify the numbers of instructors who did not support the use of multiple choice questions in summative assessment because in the survey we did not ask such question. We acknowledge this group of instructors as they provide comments that they did not use multiple choice questions in summative assessment in the open-ended column for the particular question and we have quoted these in our qualitative data tables. Some of them possibly either refused to answer our survey or omitted those questions relating to multiple choice questions; we recorded these responses as no opinion.
rather than a neutral response.

On the other hand almost half of the responses posit that students think carefully before selecting the best answer for multiple choice questions. In addition we find that the almost half of the responses believe that multiple choice questions do test student understanding of programming concepts. This would support our findings in Chapter 5 that provide strong evidence that multiple choice questions in summative assessment are meant to test low levels of Bloom’s Taxonomy. We showed that multiple choice questions tested only the lowest three levels of Bloom’s Taxonomy in the summative assessment and these questions were not all found to be easy by all students.

Some of the instructors who responded to this survey claim that multiple choice questions are regarded as easy questions as their reason to use multiple choice questions in assessment. However, based on the responses to the multiple choice questions in the summative assessment for Programming 1 in semester 2, 2007, only 55.5% were correct on an average, for 16 multiple choice questions, with a standard deviation of 15.18. In other words, on average, 56 out of 100 students had the probability of getting the correct answer for the multiple choice questions. For semester 1, 2008 the figure increased to 66.3% with a standard deviation 17.76. The standard deviation is a measure of the variability of a data set. Both of the semester standard deviations can be considered high, indicating that the data is spread over a large range of values. Hence, there can be no doubt that the students did not all find the multiple choice questions to be easy. For example, for Question 19 there were 33% of correct responses from 220 students of Semester 1, 2008 for Programming 1 course. Only one third of the 220 students managed to choose the correct answer. A multiple choice question should not be labelled as an easy question, as if the questions and distractors are well designed, it can be difficult for novices to find the correct answer.

In the final section of the survey the instructors evaluated the questions based on the criteria provided. The mean for Questions 8 indicate that the question is in the low level of instructor levels of complexity.

Question 9 has been acquired from the BRACElet set and the benchmark given by the BRACElet group is 62%, based on the student performance in summative assessment. The percentage of the correct responses for this question is in the medium level of novice levels of difficulty (based on classification stated in Section 5.4.3). The mean for Questions 9 indicates that the question is in the medium level of instructor levels of complexity. Question 10 is also in the medium level of instructor levels of complexity categorised based on the mean of the four criteria.
CHAPTER 6. INSTRUCTOR PERSPECTIVES OF MULTIPLE CHOICE QUESTIONS

Question 11 is a short answer question and so there is no answer given and no distractors are provided. The instructor responses for this question showed that they rated the multiple choice questions to be of the same instructor levels of complexity or higher based on our four criteria: syntax knowledge, semantic knowledge, problem solving skills and level of difficulty. This could be interpreted as test instrument itself does not portray the level of the knowledge or skill being tested in a question. Therefore, we cannot simply say in general that multiple choice questions are easy, or only test low levels of understanding. They can be modified to be medium or even high level of complexity if they are well designed as suggested by one of the instructor from the survey.

Currently, there are no specific criteria for instructors to use when they devise questions for programming assessments. Usually questions for summative assessments are selected according to their suitability and coverage of topics. This is supported by majority of the instructors from the survey who agreed or strongly agreed that in the summative assessment, the question needs to cover a particular programming topic and the questions need to have a certain difficulty of problem (refer Table 6.2). Therefore we would like to propose the four criteria of instructor levels of complexity to be considered by instructors when they devise questions.

6.7 Summary

Our findings highlight that almost half of the instructors we surveyed believe that summative assessment is, and is meant to be, a valid measurement of a student’s ability to program. Half of the instructors further believe that multiple choice questions provide a means of testing a low level of understanding and some added qualitative comments to suggest that multiple choice questions are easy questions and very few mentioned that they refused to use them. There was no agreement around the proposition that if a question was designed to test a low level of skill, or a low level in a hierarchy of a body of knowledge, that such a question should or would be found to be easy by students in their assessment.

We introduced four criteria for the instructor levels of complexity: syntax knowledge, semantic knowledge, problem solving skills and level of difficulty to aid our analysis in assessing questions, specifically for multiple choice questions. Given the support from instructors for the usage of multiple choice questions in summative assessment, we are strengthened in our resolve to develop a learning tool, as outlined in the Introduction chapter, with a view to encouraging novices to practise simple programming problems using multiple choice questions.
as the test instrument. This will help to identify possible misconceptions about programming in the very early part of the semester.

In addition, we find that the almost half of the responses believe that multiple choice questions do test student understanding of programming concepts. Hence, our proposed guided learning tool may benefit the usage of multiple choice questions as the test instrument as we aim to create questions to guide novices in the *remember* and *understand* levels of Bloom’s Taxonomy.

Such a learning tool will hopefully better inform instructors about novice learning difficulties in programming and allow for early intervention to help problematic students. Moreover, the proposed learning tool would potentially allow for a better design of multiple choice questions, that can be use as a part of the summative assessment. In the next chapter we discuss the learning approach to aid novices learn programming in detail. We will explain the classifications of the questions and the learning theories we apply in order to develop a learning tool based on the guided learning approach.
Chapter 7

Guided Learning Tool for Novice Programming

In Chapter 3 we highlighted that every individual is unique and learns at their own pace. Plus, students’ engagements in the tutorials are better compared to the lectures session. Accordingly, we believe that personal attention from instructors provides the best encouragement for programming students. However, this is not possible in a class that has a large number of students, so the learning materials must be flexible to suit human variability.

In this chapter we explain how multiple choice questions may be incorporated into learning experiences of novices based on Bloom’s Taxonomy. We have studied the value of employing multiple choice questions as test instruments to aid learning of programming by novices (Chapters 2, 5 and 6). We propose a framework that supports a guided learning approach using multiple choice questions. We aim to aid learning of fundamental concepts of Java programming using this approach.

Therefore, we structure this chapter as follows. In Section 7.1 we explain our motivation to propose a guided learning approach and implement the guided learning tool to support the approach. The guided learning tool was developed during two cycles of action research (cycles 4 and 5). The cycles represent the development process, incorporating novice responses to the tool and our reflections and discussions on improving the guided learning approach and tool.

In Section 7.2 we present and discuss the task and the questions included in the guided learning tool developed in Cycle 4 of the action research methodology, and alternatively labelled as guided learning A. We present these questions in two sections: Section A and
Section B. We discuss the questions and their relationships to Bloom’s Taxonomy.

In Section 7.3 we describe the questions included in guided learning B developed in Cycle 5 of action research methodology. Most importantly, we present our proposed model of guided learning programming based on Bloom’s Taxonomy in this section.

7.1 Motivation: A Guided Learning Approach to Aid the Learning of Programming

The traditional delivery model for teaching programming is not ideal because programming is best absorbed in one-on-one contexts [Merrill et al., 1995], by self practice [Eckerdal, 2009], brainstorming with others, or indeed in face-to-face discussions with teachers or mentors and students [D’Souza et al., 2008]. We believe that conceptual difficulties are better clarified with interaction, during which many conceptual gaps are closed or clarified, enabling the learner to build confidence and to do so at the early, critical stages of their learning experiences. Also, the traditional delivery model requires the students to be on campus at specific times in order to attend lectures, tutorials and laboratory exercises, which can disrupt an individual’s learning pace, thought processes and self-practice required to really learn something deeply.

Novices often have difficulty in program design, therefore, our research focuses on aiding novices who are having difficulties in program design either via problems requiring translation of questions into valid programs or interpretation of programs. We approach these learning difficulties by decomposing the tasks embedded in a program. We aim to aid novices by guiding them to establish proper mental models of fundamental programming concepts. We define a proper mental model of a programming concept to be at the relational level of SOLO Taxonomy (represented in Table 3.2), whereby most of the components in a task are recognised as being related and linked. In a relational level novice make sense of each component and respond to the general idea of the task required to better reflect what does happen at the relational level. The relational level may suit the learning objective that requires novices to link most of the components they have learnt. If students in the Programming 1 course level are able to achieve the extended abstract level, this would be considered a bonus, because it would mean that such novices would be able to generalise their knowledge and extend it into a new domain.

Our aim is to guide the students through the relational level to the multistructural level of the SOLO Taxonomy. We arrange the questions in the guided learning tool based on Bloom’s Taxonomy (see Figure 7.1) to enable the novice to reach at least the multistructural
We focus our study on the early learning stages (first four weeks of a semester) of introductory programming courses. We consider the first four weeks of a semester learning introductory programming course to represent the most crucial stage to encourage novices to sustain their interest in learning to program because the “metaknowledge” may increase engagement and motivation and it is important to keep up with the flow of newly introduced concepts [Robins, 2010]. Novices who have low motivation and disengage should be followed up as early as the first week.

The guided learning tool was developed with a dual purpose. The first is to provide a “practise instrument” to allow students to explore and consolidate their knowledge of programming. The second aim is to try to understand or probe the minds of the novices programmers to enhance the practice environment with a learning emphasis that attempts to identify cognitive gaps. Our work aims to help students resolve some of their early misconceptions and improve their understanding of introductory programming concepts. We propose a set of programming exercises within our guided learning tool, in order to aid novice programmers. These exercises are based on the levels of difficulty aligned with Bloom’s Taxonomy. We apply the Bloom’s Taxonomy and the instructor levels of complexity reported in previous chapters. We classify and arrange the questions in the guided learning tool based on the learning taxonomies to guide novices to explore learning fundamental concepts of programming. Student responses to the questions may reflect their understanding of the particular programming concepts. Hence, the guided learning tool may assist instructors to identify strong and weak areas of learning to program. Instructors may monitor the performance of novices and emphasise specific topics or areas where novices have difficulties.

A personalised learning approach supports the variation in the capabilities of novices to learn programming. Novices can study at their own pace. We want the exercises to be presented in increasing levels of cognitive difficulty. Novices may also self-navigate to explore the exercises up to the level of difficulty that suits their learning capability. Some novices may take their time to understand a concept, while others may progress to the questions that require higher skills faster. We also adopt the idea of using a guided learning approach to encourage novices to guide themselves through the exercises and find a balance between deep and surface learning approaches.

In other issue, usually, there is a specific time allocated for one-to-one interactions and for consultations with the programming instructors or tutors or mentors. Unlike those traditional systems, the guided learning tool can be used any time and anywhere via Internet connection.
Novices may also use it repeatedly to establish their skills to program. They need not feel de-motivated or ashamed to do the exercises many times as they do not have to face the instructors or mentors perceptions of not meeting their levels of expectations.

Other than that, the guided learning tool has incorporated some of the more common known errors and misconceptions identified through the experiences of teaching staff and in the existing literature from previous research. Novices may learn through commonly errors made [Kummerfeld and Kay, 2003]. We employ some of the questions in the guided learning tool from PeerWise (further descriptions in Section 2.3.1), which is a test bank of multiple choice questions largely contributed to by novices [Denny et al., 2008b]. These questions had been posted by novices for their peers and some of the questions appear to be the common errors for novices (as discussed in Section 2.5).

Overall, we hope that through the guided learning tool, novices will develop a strong foundation of programming concepts to enhance their programming skills in more advanced programming courses.

The guided learning tool has been developed and evaluated in two cycles of action research referred to as guided learning A and guided learning B. Guided learning A has two sections of questions: debugging and multiple choice questions, while guided learning B focuses on multiple choice questions. The guided learning tool was developed as a prototype to try the concepts proposed in this research.

In the next section we explain guided learning A and guided learning B and their relation to Bloom’s Taxonomy.

7.2 Guided Learning A

In this section we describe guided learning A in detail. Guided learning A is available online at: www.cs.rmit.edu.au/~sshuhida/survey2010/survey.html. We present the screenshots of guided learning A in Appendix E.

We use the revised Bloom’s Taxonomy [Anderson et al., 2001], which we discussed in Section 3.2, to classify the tasks in the guided learning A. The revised version of the Bloom’s Taxonomy has improved the categories in the original Bloom’s Taxonomy (refer Section 3.2.2). The revised Bloom’s Taxonomy is better suited to portray the classification of the questions in the guided learning tool. For example, the lowest level of the original taxonomy is knowledge whereas the revised taxonomy is remember. Based on the lowest level of the original Bloom’s Taxonomy, students will require remembering skills of a specific information whereas
the revised Bloom’s Taxonomy a question in this level may test the students’ knowledge and ability to a particular task. Therefore based on this example, the classifications of the revised Bloom’s Taxonomy is better than the original version of Bloom’s Taxonomy in terms of the keywords used to explain the task tested to students.

![Diagram of Bloom's Taxonomy](image)

**Figure 7.1: Applying Bloom’s Taxonomy to the Guided Learning A**

In Figure 7.1 we explain the application of the revised Bloom’s Taxonomy to **guided learning A**. There are two sections in **guided learning A**, which we label as Section A: Debugging Question, which requires novices to detect and fix the errors; and Section B: Multiple Choice Questions that follow up from a sample work of the **Hello World** program (refer to Figure 7.1).

Section A contains learning through debugging questions which cover the higher levels of Bloom’s Taxonomy, apply and analyse, as learners need to analyse the program to detect the errors and apply their knowledge to fix the errors and rewrite the correct code. Section B contains multiple choice questions, which require the two lower levels of Bloom’s Taxonomy, remember and understand of fundamental programming concepts.

Since this is intended for an introductory programming course, we do not test novices at the highest levels of Bloom’s Taxonomy, evaluate and create. At the evaluate level, students are required to assess other people’s work to achieve this level and at the create level students are expected to produce a complex programming project with a large scope.

In the following subsections we demonstrate sample questions for the two sections of **guided learning A**.

### 7.2.1 Section A: Debugging Questions

Debugging is a process to analyse, detect and fix programming errors [Ramalingam and Wiedenbeck, 1997]. In line with Lapidot and Hazzan, we believe that learning to debug a
program can improve and build a student’s mental model to program computer [Lapidot and Hazzan, 2005].

The question in this debugging section tests the ability to detect and fix the syntax errors. This section requires a high level of cognitive skill as novices need to be able to read the programming code and analyse it to detect any errors. Novices may require the analyse skill to be able to detect the error and can then apply their knowledge to write the correct program (refer Figure 7.2).

Figure 7.2 presents an example of a debugging question. The program will print Hello World after all the errors in the program are fixed. There are a few syntax errors in the program and three tasks in this type of question. First, novices need to identify the errors by ticking the boxes for each erroneous line. Second, novices are required to state the total number of errors. Third, they are required to write the correct program.

7.2.2 Section B : Multiple Choice Questions

This section of guided learning A highlights the main keywords associated with questions in regards of the syntax of the program. We aim to help novices decompose the components of a program and guide them by undertaking each component one at a time.

We believe novices struggle to remember and understand the jargon in the Java language (refer Chapter 2). During the first few weeks many definitions, terminologies and symbols are introduced to novices for them to absorb, recognise and learn. Hence, the questions in this section require novices to use the remember and understand skills of Bloom’s Taxonomy. Although these are the two lowest levels in Bloom’s Taxonomy, we cannot assume that this is easy. The questions in this section expose novices to the terminologies and symbols in the Java language and their relevance to the particular program.

We discuss one example for this type of question. We use the Hello World program (Figure 7.3) is usually introduced to the novices by their instructors in the first week of the semester in introductory programming course. It is a program wherein novices are required to print Hello World. Each hyperlink in Section B is chosen from the keywords in the program and will be linked to a multiple choice question, which in turn have difficulty levels based on Bloom’s Taxonomy. As an example, by clicking on the import java.io.*, novices are linked to the questions at the remember and understand levels of Bloom’s Taxonomy.

For the multiple choice questions section, novices are presented with a program, similar to the one that novices attempted to correct in the debugging question (Section 7.2.1). There
CHAPTER 7. GUIDED LEARNING TOOL FOR NOVICE PROGRAMMING

SECTION A: Detect and fix the program given

1. Please detect the error(s) by ticking the check box of the line contain error(s).

The program below has error(s). OUTPUT of the given Java Hello World Example would be:

Hello World

1. import java.io.*;
2. /*
3. Java Hello World example.
4. /
5. public class HelloWorld
6. {
7. public static void main(string [])
8. {
9. System.out.println("Hello World !")
10. }
11. }

Total number of error(s):

2. Now, fix the program above:

```java
import java.io.*;
/*
Java Hello World example.
*/
public class HelloWorld
{
    public static void main(string [])
    {
        System.out.println("Hello World !")
    }
}
```

Figure 7.2: Example of Debugging Question, Section A in Guided Learning A
CHAPTER 7. GUIDED LEARNING TOOL FOR NOVICE PROGRAMMING

Figure 7.3: Example of Multiple Choice Question, Section B in Guided Learning A
are seven multiple choice questions and, for each one, novices must choose the correct answer from a list of options provided. For questions 1 to 7, only three options were given, which were A, B and C, except for Question 5, which required novices to select between yes or no.

Details of each multiple choice question are as follows.

**Question 1**

Question 1 requires novices to recall the definition *input* and *output*. Recall is classified in the *remember* level of Bloom’s Taxonomy.

What does io stand for?
A. in out
B. input output
C. I do not know the answer for this question

**Question 2**

As a follow up to Question 1, Question 2 tests the novices on the *understand* level of Bloom’s Taxonomy by asking for the usage of the *input* and *output* package. It requires remembering definitions of *io* but also understanding how *io* works.

What is the purpose of import java.io.*; ?
A. To make use of the method(s) of the library in other systems
B. To import the input and output of the program to other systems
C. I do not know the answer to this question

**Question 3**

Question 3 tests novices on the symbol that is used to take statements out of a program. All the words written between /* and */ will not be read by the compiler.

What is the purpose of /* . */ ?
A. To comment out statement(s)
B. To terminate a statement
C. I do not know the answer to this question

148
CHAPTER 7. GUIDED LEARNING TOOL FOR NOVICE PROGRAMMING

Question 4
As a follow up to Question 3, Question 4 tests novices ability to understand and differentiate between // and /* */.

What is the difference between /* .. */ and //?
A. /* .. */ is used to comment out a line and // is used to comment out a block of statement
B. /* .. */ is used to comment out a block of statement and // is used to comment out a line
C. I do not know the answer to this question

Question 5
Question 5 requires novices to know the appropriate access control for each of the modifiers. Novices must choose between yes or no access controls for the modifiers.

Question 5: Choose the appropriate access control for a program based on its modifier.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>private</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>default</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>protected</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 6
Question 6 tests knowledge about the usage of the semicolon (;) symbol. Novices frequently misuse semicolons, which ends up in failure to compile the program.

What is the purpose of a semicolon (;) in a program?
A. To comment out a line
B. To terminate a statement
C. I do not know the answer to this question

Question 7
Question 7 also looks at a frequently used symbol in a computer program. It tests the definition of args.
What does args stand for?
A. arguments
B. agree
C. I do not know the answer to this question

7.3 Guided Learning B

*Guided learning B* is the improved version of *guided learning A*. In this version we use multiple choice questions to test on *remember* and *understand* levels of Bloom’s Taxonomy and we do not provide the debugging exercise for novices. We decide to add the questions for novices to *apply* their knowledge of programming. The rationale for this decision is as follows:

- We want to focus on the three lower levels of Bloom’s Taxonomy; *knowledge*, *understand* and *apply*, which are more critical learning skills and should be emphasised during the first four weeks of the Programming 1 course. Moreover based on our analysis of the multiple choice questions in Programming 1’s summative assessment, the questions are classified in the three lower levels of Bloom’s Taxonomy (refer Section 5.4.1).

- The task of debugging code is classified in the *analyse* level of Bloom’s Taxonomy. Therefore we decided to omit debugging questions in guided learning B to focus on to strengthen the lower levels of Bloom Taxonomy. Furthermore debugging is a difficult task. We do not want our exercises to discourage novices to learn programming.

We want to aid novices to construct proper mental models in developing a computer program. We employ a bank of questions as a guide in learning based on the lower levels of the cognitive domain of Bloom’s Taxonomy. *Guided learning B* provides more multiple choice questions compared to *guided learning A*. It is intended for the first four weeks of learning programming in Programming 1 course, which usually includes coverage of basic syntax, operators, selection and iteration topics. It is available online for easy access for novices and is available at: www.cs.rmit.edu.au/~sshuhida/DG/session/login.php. We present the screenshots of *guided learning B* in Appendix F.

We develop a framework that supports the application of Bloom’s Taxonomy to our *guided learning B*. Figure 7.4 explains the framework of our guided learning tool and its relationship to Bloom’s Taxonomy. As mentioned previously, *guided learning B* focuses on the three lower levels of Bloom’s Taxonomy: *knowledge*, *understand* and *apply*. We divide the exercises into three sections based on the learning taxonomy. We provide multiple levels of exercises to
CHAPTER 7. GUIDED LEARNING TOOL FOR NOVICE PROGRAMMING

Figure 7.4: Applying Bloom’s Taxonomy to the Guided Learning B

scaffold the learning of programming. We hope that this approach will help to decrease the cognitive load.

The three sections are distinguished as follow: Section A, which provides a sample program for novices to study; Section B, which provides hyperlinks from the keywords in Section A to further follow-up the multiple choice questions; and Section C, which requires novices to apply their skills to write a complete program.

7.3.1 Section A

In Section A of the guided learning tool, a basic sample program will be provided to novices. Novices can scan through a working program to become familiar with the general idea of a
basic working program that demonstrates the fundamental learning concepts. We label this section as the “task”. The task will be at the centre of the framework (refer to Figure 7.4). Hyperlinks will be provided to emphasise the importance of the keywords and these hyperlinks will be connected to Section B of the guided learning tool. In the sample below, the hyperlinks are in blue font.

Sample Task 1:
import java.io.*;
// Java Hello World example.
public class HelloWorld
public static void main (String args[]) {
    System.out.println("Hello World");
}

7.3.2 Section B

In Section B, referring to Figure 7.4, each ring represents our classifications of the questions based on the Bloom’s Taxonomy. The hyperlinks in Section A are linked to the multiple choice questions in Section B, which cover the remember and understand levels of Bloom’s Taxonomy. These multiple choice questions have various levels of difficulties and we arrange them from the lowest to the highest level of difficulty. Novices may self-navigate these questions according to their interest in learning.

Referring to the example in Figure 7.5, the task shows an example of the Hello World program. We scaffold the learning by having different layers of the exercises based on the classifications of Bloom’s Taxonomy. As an example, the import java.io.* line, when clicked, will require novices to recall the io terminology, represented in the remember level of Bloom’s Taxonomy. Next, novices are required to answer the purpose of import java.io.*.

We present another example of the questions in the remember and understand levels of Bloom’s Taxonomy that are linked from the task provided in Figure 7.6. The questions require novices to remember and understand the usage of commenting out lines of programming code.
Chapter 7. Guided Learning Tool for Novice Programming

Figure 7.5: Week 1 Content of the Guided Learning B: Question 1

**REMEMBER:**
What does io stand for?
A. in out
B. input output
C. I do not know

**UNDERSTAND:**
What is the purpose of `import java.io.*`?
A. To make use of the method(s) of the library in other system
B. To import the input and out of the program to other system
C. I do not know

```java
import java.io.*;
//Java Hello World example.
public class Hello {
    public static void main (String args[]) {
        System.out.print
        ("Hello World");
    }
}
```
Figure 7.6: Week 1 Content of the Guided Learning B: Question 2
We noted that some acronyms may seem irrelevant in learning programming, but we firmly believe that the novices need to become accustomed to the use of common acronyms and the definitions of keywords in programming as this will facilitate progress to the next level of Bloom’s Taxonomy, that is, to understand the subject matter. The Java programming language is based on the English language. Some of the jargon may translate directly into English. Some may work as an acronym, such as io for input-output, args for arguments and ln for line.

### 7.3.3 Levels of Complexity for Multiple Choice Questions

One level of Bloom’s Taxonomy may have multiple levels of task difficulties. We use our proposed instructor levels of complexity (syntax knowledge, semantic knowledge, problem solving skills and level of difficulty) criteria to arrange the multiple choice questions in guided learning B. These measures have been discussed in Chapter 6. There are three levels for each criterion, one for low, two for medium and three for high. We discuss an example of organising the questions in the multiple choice questions section. The example involves Question 12 and Question 20 from the guided learning B.

**Question 12**

Question 12 requires novices to understand the skills used to swap between the three variables (variable a, b and c). Syntactically, the code is easy to read, therefore it requires a low level of syntax knowledge. The key challenge to this question is the interpretation of the language, including which variable should be assigned to which identifier, and this requires a medium level of semantic knowledge, and so we rate as a low problem solving skills and a medium level of difficulty. The question may suit to be tested as a benchmark to test the novices knowledge as the swapping variable task is the simplest piece of non-iterative code that requires the same form of relational reasoning as iterative code [Corney and Teague, 2011]. Therefore, if novices struggle to swap variables, then the beginning of the semester of learning introductory programming does not prepare novices well to learn iteration. Hence clearing the misconception (if any) in the beginning of the semester may aid student to progress to the next learning programming task smoothly. If there are students who struggle with the code for swapping two variables, then the early part of semester does not prepare students well for the iterative code to follow.
Question 12: What is the output of the following code?

```java
int a = 5;
int b = 3;
int c = 10;
a = c;
c = b;
b = a;
System.out.println(a + " " + b + " " + c);
```

A. 10 10 3  
B. 5 3 10  
C. 10 3 5  
D. 10 3 10  
E. 3 5 3

**Question 20**

In Question 20 novices are tested on data types and arithmetic operators. This question requires novices to calculate the numbers given and provides the answer in the format of an integer. We rate this question as a low level of syntax knowledge. As in mathematics, novices need to know that in computing with integers, the operations of multiplication, division and modulus (remainder of a division) must be conducted before addition and subtraction and the answer must be an integer. Therefore, we rate this question as a medium level of semantic knowledge, a low level of problem solving skills and a low level of difficulty.

Question 20: What value will be assigned to the variable x as a result of the following statement?

```java
int x = 10 / 4 + 10 % 4 * 2;
```

A. 0  
B. 4  
C. 6  
D. 8  
E. None of the above

We compare Question 12 and Question 20 in Table 7.1. On average Question 12 is rated...
higher than Question 20. Therefore, we decided to place Question 20 before Question 12 in the multiple choice questions section of the guided learning B.

### 7.3.4 Section C

In Section C novices are required to apply their knowledge to write a complete program. A sample of the types of question in Section C is shown below.

Based on the snippet of the program in Section A, write a program that will ask the user to enter his/her name and print out Hello and the entered name.

The example of the print out is as below:

Please enter your name: Mr Happy
Hello Mr Happy

### 7.3.5 Guided Learning B: Getting It All Together

In previous subsections we discussed the three sections in guided learning B. In this subsection we explain the flow of the questions in the guided learning B in Figure 7.7. Each ring represents a level of difficulty of the question and each petal represents a component that novices must construct in order to learn programming.

Each terminology in the computer program from Section A may initiate a question or more than one question in the remember and understand levels of Bloom’s Taxonomy in Section B. If there is more than one question in these two levels of Bloom’s Taxonomy, the questions will be arranged based on the instructor levels of complexity. In Section C, novices are required to apply their knowledge to write a complete program. Novices will build up their knowledge by completing the exercises from Section A to Section B then Section C.

This framework may allow instructors to monitor novices weaknesses. They can check the answers given by novices and find out whether or not they were able to answer the question.
correctly, in the context of a specific section and specific concept. Instructors may highlight the particular programming concept in their teaching to help novices overcome their learning difficulties around that particular concept. Hence, this framework facilitates the idea of probing the novices minds to understand their learning barriers in programming.

Figure 7.8 and figure 7.9 are screenshots of guided learning B. The screenshots show the three different sections of guided learning B. Section A is the sample program, the questions
in Section B raise the level in Bloom’s Taxonomy from *remember* to *understand* (figure 7.8 and figure 7.9). In Section C novices are required to write a program.
**Guided Learning Tool**

- **Basic Syntax**
- **Operators**
- **Selection**
- **Iteration**
- **Other**
- **Log Out**

---

**Learning objective:** You are exposed to basic syntax and the general idea of how a program works.

**Instructions:** Click on the hyperlinks in **SECTION A** for more questions in **SECTION B** to aid you in learning about the particular subtopic.

---

**SECTION A**

The following program prints out:

```
Hello World
```

1. `import java.io.*;`
2. `//Java Hello World example.`
3. `public class HelloWorld`
4. `{`
5. `public static void main(String args[])
6. `{`
7. `System.out.println("Hello World");`
8. `}
9. }
10. }
```

---

**SECTION B:** Select the answer from the options given.

1. What does it stand for?
   - (A) input
   - (B) output
   - (C) I do not know the answer for this question

---

**SECTION C:** Advanced exercise for you to try:

Based on Question 1, write a program that will ask the user to enter their name and print out:

```
Please enter name: Mr.Happy
Hello Mr.Happy
```
CHAPTER 7. GUIDED LEARNING TOOL FOR NOVICE PROGRAMMING

SECTION A
The following program print out:

```java
1. import java.io.*;
2. public class HelloWorld
3. {
4.     public static void main(String args[])
5.     {
6.         System.out.println("Hello World");
7.     }
8. }
```

SECTION B: Select the answer from the options given.

2. What is the purpose of import java.io.*;
   - A. To make use of the method(s) of the library in other system
   - B. To import the input and output of the program to other system
   - C. I do not know the answer for this question

SECTION C: Advance exercise for you to try.
Based on Question 1, write a program that will ask the user to enter his/her name and print out:

```java
Please enter name: Mr. Happy
Hello Mr. Happy
```

Question 2: Test on novices ability to Understand levels of Bloom Taxonomy
CHAPTER 7. GUIDED LEARNING TOOL FOR NOVICE PROGRAMMING

7.4 Summary

In this chapter we have discussed the process of developing our guided learning tool. We implemented two cycles of action research to establish the development of the learning tool. We labelled the two version of the guided learning tool as guided learning A and guided learning B. The guided learning A has two sections which consist of Section A (debugging question) and Section B (multiple choice questions).

We introduced guided learning B as an improved version of the guided learning tool. In guided learning B, we make use of Bloom’s Taxonomy to decompose tasks according to the level of cognition required by the question. Within a given level of Bloom’s Taxonomy, we found that a question may have a range of difficulties. Therefore, we use our instructor levels of complexity as introduced in the Chapter 6 to measure the complexities of the multiple choice questions in Section B of guided learning B.

As stated in the preamble of this chapter, each cycle of action research represents the development process, incorporating novice responses to the tool and our reflections and discussions on improving the guided learning tool. We have presented the development process of the guided learning tool in this chapter. Hence, in the next chapter we discuss our findings and reflections of the guided learning tool in the two cycles of action research.
Chapter 8

Evaluation of Guided Learning Tool

In Chapter 7 we explained the theories underpinning the development of our guided learning tool. The guided learning tool helps novices by providing multiple levels of exercises that, hopefully, will help them to decompose or break down the steps in learning to program. We also discuss some of the questions from the sections in the two versions of the guided learning tool, guided learning A and guided learning B.

In this chapter, we present the evaluation of our guided learning tool. Novices voluntarily trialled both versions of the guided learning tool. They completed a survey to provide their opinions about the guided learning tool. The survey solicited their perceptions about the learning tool and their suggestions for improving it. We evaluate the guided learning tool in two cycles of action research (cycles 4 and 5), which involve two different groups of novices, the first group trialling guided learning A and the second group trialling guided learning B. Our reflections over the guided learning A helped to focus the improvement for the development of guided learning B.

This chapter is structured as follows. In Section 8.1 we explain the survey designed to evaluate the guided learning tool, (guided learning A and guided learning B). We present the data collected in two sections; responses to guided learning A and guided learning B. In Section 8.2 we discuss the data collected and the suggestions by novices to improve the guided learning A.

In Section 8.3 we highlight our reflections about guided learning A to provide improvements in developing guided learning B. In Section 8.5 we present our findings for guided learning B. In Section 8.6 we discuss the survey responses for guided learning A and guided learning B. In Section 8.7 we compare the final grades of overall assessments for novices who
used the guided learning tools to those who did not. Last but not least, in Section 8.8 we discuss our findings.

8.1 Surveys for the Guided Learning Tools

The evaluations for the guided learning tool were conducted over two cycles of action research (Cycle 4 and Cycle 5). In Cycle 4 we approached first year novices enrolled in our introductory programming course, Programming 1. In Cycle 5 we approached novices enrolled in Java for Programmers course. Detailed course descriptions are presented in Table 4.7.

Both courses (Programming 1 and Java for Programmers) use the Java language as the teaching vehicle. Java for Programmers may not be a student's first course in programming. Students enrolled in Java for Programmers may have learned or used, C or PHP language, but have no experience with the Java language. We considered them as novice programmers because they meet the criteria we set in Section 4.8.1, they are individuals who enrolled in university classes and learning fundamental concepts of programming. Furthermore the guided learning tool may benefit them in learning programming for first four weeks of a semester. Students in the Java for Programmers course learn syntax and basic usage of the Java language (including sequence, selection and iteration) in the first four weeks and so it is relevant for them to apply the guided learning tool. We accept that this group of students may have a better logic knowledge, but, the questions in the remember and understand levels of Bloom's Taxonomy emphasise the syntax knowledge of Java languages, and this may benefit this group of students.

In order to promote our surveys we sent emails to the novices via the course coordinator. We also approached novices in lectures and in scheduled computer laboratory classes. Participation was voluntary and novices were able to access the guided learning tool at any time throughout the semester. We collected the data through online surveys.

We compared four cohorts of novices from the two programming courses. The demographic data for the respondents are shown in Table 8.1. The first two groups were the novices who enrolled in Programming 1 course. We identified these groups as P1G1 and P1G2. P1G1 denotes the novice cohort from semester 2, 2009, and P1G2 the novice cohort from semester 2, 2010.

The next two groups were the novices enrolled in Java for Programmers course. We identified these groups as JPG1 and JPG2. JPG1 is the group of novices who took the course during the summer semester of 2010, and JPG2 denotes the group of novices enrolled
CHAPTER 8. EVALUATION OF GUIDED LEARNING TOOL

Table 8.1: Respondent Profile

<table>
<thead>
<tr>
<th>Descriptions</th>
<th>P1G1</th>
<th>P1G2</th>
<th>JPG1</th>
<th>JPG2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Enrolled</strong></td>
<td>Programming 1</td>
<td>Programming 1</td>
<td>Java for Programmers</td>
<td>Java for Programmers</td>
</tr>
<tr>
<td><strong>Semester</strong></td>
<td>2</td>
<td>2</td>
<td>Summer</td>
<td>Summer</td>
</tr>
<tr>
<td><strong>Year</strong></td>
<td>2009</td>
<td>2010</td>
<td>2010</td>
<td>2011</td>
</tr>
<tr>
<td><strong>Number of novices</strong></td>
<td>141</td>
<td>221</td>
<td>33</td>
<td>32</td>
</tr>
<tr>
<td><strong>Level:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undergraduate</td>
<td>139</td>
<td>205</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>Postgraduate</td>
<td>2</td>
<td>16</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td><strong>Gender:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>117</td>
<td>189</td>
<td>27</td>
<td>26</td>
</tr>
<tr>
<td>Female</td>
<td>24</td>
<td>32</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

in Java for Programmers during the summer semester of 2011.

We chose to trial the guided learning tool to P1G2 and JPG2 groups. The control groups are those who did not trial the guided learning tool. P1G1 and JPG1 are the control groups while groups P1G2 and JPG2 are the ones who trialled the guided learning tool. We want to have the two groups because we will compare their performance at the end of the semester for their respective courses.

In the next section we report the results of evaluations by novices of guided learning A and guided learning B.

8.2 Responses to Guided Learning A

We received 11 responses from the P1G2 cohort. Given that there were 221 novices enrolled to the Programming 1 course for this particular semester, we acknowledge the very low response rate of only 5% of the class; however, we felt that these responses would be of value as even a small number would potentially be beneficial to us in order to improve the learning tool for future use. The P1G2 cohort consisted of full time students, which in an average they enrolled to 5 or 6 courses in a semester, this may be the reason that they were not able
CHAPTER 8. EVALUATION OF GUIDED LEARNING TOOL

to commit to using the guided learning A.

Furthermore, participation was voluntary, so these views of novices were highly likely to be sincere and their willingness to contribute to the body of knowledge of computer science education field should not be disregarded.

In Chapter 7 we explained that the guided learning A consists of two sections; debugging question and multiple choice questions. In the next section we detail our findings collected from novices.

8.2.1 Section A: Debugging Question

For the debugging question, novices were presented with eleven lines of code. Novices were required to identify the erroneous lines and subsequently fix the errors. The program given is as below:

1. import java.io.*;
2. /*
3. Java Hello World example.
4. */
5. public class HelloWorld
6. {
7.     public static void main(string []
8.     {    
9.         System.out.println("Hello World !")
10.     }
11. }

There are five syntax errors in the program. We denote the first error as $e_1$, second error as $e_2$, third error as $e_3$, fourth error as $e_4$ and fifth error as $e_5$. The correction of $e_1$ requires novices to place a “*” symbol before the “/” symbol in line four. Eight out of 11 novices detected and fixed this error.

The second error correction ($e_2$) requires novices to add the missing “c” in “public”. Nine novices managed to detect and fix this error. Next, there are two errors in line seven. The third correction to $e_3$ requires novices to change the letter “s” to its capitalized equivalent “S”, for the word “String”. Only two novices detected and fixed this error. Eight novices were able to detect and fix $e_4$ by adding the word “args” on line seven. The next error, $e_5$ is
the missing semicolon (;) on line nine. Eight novices managed to correct this error. Overall, only one novice managed to detect and fix all the errors in the program.

### 8.2.2 Section B: Multiple Choice Questions

In this section we outline the responses by novices to the multiple choice questions in Section B of guided learning A. We have discussed these questions previously in Section 7.2.2.

**Table 8.2: Novice Responses to Multiple Choice Questions**

<table>
<thead>
<tr>
<th>Question Number</th>
<th>Option A</th>
<th>Option B</th>
<th>Option C</th>
<th>Correct Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.1% (1)</td>
<td>90.9% (10)</td>
<td>0</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>54.5% (6)</td>
<td>45.5% (5)</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>100% (11)</td>
<td>0</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>90.9% (10)</td>
<td>9.1% (1)</td>
<td>B</td>
</tr>
<tr>
<td>6</td>
<td>9.1% (1)</td>
<td>90.9% (10)</td>
<td>0</td>
<td>B</td>
</tr>
<tr>
<td>7</td>
<td>100% (11)</td>
<td>0</td>
<td>0</td>
<td>A</td>
</tr>
</tbody>
</table>

Table 8.2 reports responses by novices to multiple choice questions in Section B of guided learning A. Question 3 and Question 7 require novices to recall basic knowledge of programming. In Question 3 novices are tested on the purpose of comments “/* .. */” and Question 7
test the meaning of “args”. All novices who trialled guided learning A identified the correct answers for these two questions. For Questions 1, 4 and 6 only one novice did not select the correct answer. However, it was not the same individual who did not identify the correct answer for all three questions.

For Question 4 one novice wrote that they did not know the answer. For Question 2 six novices chose the correct answer while five chose option B, most probably because of the inclusion of the term “input-output”. We present our overall findings about the novice responses in multiple choice questions in Figure 8.2.

Question 5 asked novices about the access modifier in Java programming language. We present the correct responses highlighted in yellow (refer Table 8.4). Note that the public access modifier received the most correct responses compared to the other three types of access modifier (private, default and protected).

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>90.9 (10)</td>
<td>81.8 (9)</td>
<td>81.8 (9)</td>
<td>90.9 (10)</td>
</tr>
<tr>
<td>private</td>
<td>81.8 (9)</td>
<td>45.5 (5)</td>
<td>27.3 (3)</td>
<td>45.5 (5)</td>
</tr>
<tr>
<td>default</td>
<td>63.6 (7)</td>
<td>81.8 (9)</td>
<td>54.5 (6)</td>
<td>54.5 (6)</td>
</tr>
<tr>
<td>protected</td>
<td>90.9 (10)</td>
<td>63.6 (7)</td>
<td>72.7 (8)</td>
<td>45.5 (5)</td>
</tr>
</tbody>
</table>

Referring to 8.4 almost all novices who trialled the guided learning A knows the access
Table 8.4: Individual Correct Responses to Question 5

<table>
<thead>
<tr>
<th>Respondent</th>
<th>Correct Responses (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novice1</td>
<td>81.3</td>
</tr>
<tr>
<td>Novice2</td>
<td>62.5</td>
</tr>
<tr>
<td>Novice3</td>
<td>50.0</td>
</tr>
<tr>
<td>Novice4</td>
<td>56.3</td>
</tr>
<tr>
<td>Novice5</td>
<td>62.5</td>
</tr>
<tr>
<td>Novice6</td>
<td>62.5</td>
</tr>
<tr>
<td>Novice7</td>
<td>100</td>
</tr>
<tr>
<td>Novice8</td>
<td>50.0</td>
</tr>
<tr>
<td>Novice9</td>
<td>62.5</td>
</tr>
<tr>
<td>Novice10</td>
<td>81.3</td>
</tr>
<tr>
<td>Novice11</td>
<td>62.5</td>
</tr>
</tbody>
</table>

for public. One or two novices were not able to get the correct answer for public as the access modifier. As for the rest of the modifiers, we received mixed responses from novices, we could not identify that novices would get the correct responses wholly or partially for each of the access modifier. We analyse the responses based on the individual performance to look at the correct responses to the question in Table 8.4.

We represent Novice1, Novice2 and the others as the individual novices who trialled guided learning A. Based on the individual correct responses in Table 8.4, Novice7 is the only one who manages to answer each question correctly. Thus Novice7 is the only novice to display complete remembering skills regarding the access modifier in introductory course for Java programming. All responses managed to get more than half of the correct response. We could not identify which part of the question that novices are not able to get the correct answer, therefore we try to analyse the responses by novices to Question 5, individually and by each question, in order to understand their remembering skills with respect to the access modifier in programming.

Figure 8.3 shows responses by novices to Question 5, where the green box indicates the correct answer and the red box indicates the incorrect answer. Each row represents the subquestion’s responses and each column represents individual novice responses. Table 8.5 represents a guide to interpret Figure 8.3. As an example, referring to Table 8.5, the response for public access modifier for a class in a program is represented by a. Only Novice3 answered this part of the question incorrectly, shown as the red box in the bottom row. Although we have tried to analyse these individual responses for Question 5, it is difficult to determine whether or not the novices were guessing, as they only needed to click the key for “yes” or
“no”. Hence after this trial we are more determined to work with multiple choice questions, where we can establish a higher level of understanding of responses.

![Figure 8.3: Novices Individual Responses to Question 5](image)

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>private</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>default</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
</tr>
<tr>
<td>protected</td>
<td>m</td>
<td>n</td>
<td>o</td>
<td>p</td>
</tr>
</tbody>
</table>

We discussed the data collected from novices’ responses in guided learning A. In the next subsection we list novices’ suggestions to improve the guided learning A for us to incorporate in the next version of the guided learning tool.
CHAPTER 8. EVALUATION OF GUIDED LEARNING TOOL

8.2.3 Novice Suggestions for Guided Learning A

In the open ended questions, novices were asked for their views about guided learning A, and how it might be improved. One novice suggested that the following question be included in guided learning tool, “What’s the difference between System.out.println and System.out.print?” Other suggestions from the novices for the guided learning A included the following:

• “Teach novices how to install JDK/JRE compiler on their own computers and not to hardly rely on Eclipse or any other programming software to train their programming skills.”

• “Add basic programming examples that are similar to the exam and assignments instead of complicating examples that are not related directly to the exam and assignments.”

• “Make it more clear with regards to the questions and activities. Needs rethink of design and scope. Inconsistent site layout. Keep it simple.”

• “We need some examples of every chapter.”

• “More information for the instructions.”

• “Improve user interface.”

• “Give users more instruction.”

8.3 Reflections

We categorise the suggestions offered by novices regarding guided learning A (refer to Section 8.2.3) into three themes: instruction, examples in each topic, the webpage design. We consider these suggestions in our development of guided learning B.

8.3.1 Instruction

Based on the qualitative feedback from novices for guided learning A, we added detailed instructions in the introduction page (refer to Figure 8.4). Users are now directed to the introduction page once they have successfully read the ethics participation requirements and logged in. The instructions include explanations about how the three different sections are linked together and how each exercise has been classified.
Welcome to the Guided Learning Tool which provides exercises for learning in Java.

These exercises are intended to help you understand important concepts in introductory programming. They range from simple programs which may require you to write some basic code to more difficult exercises which require some chunks of code.

SECTION A provides a sample of a working program for you to study. There are links for new terms to remember and understand. Click on the links and a series of multiple choice questions will appear in SECTION B.

Select the best answer and press the Submit button to check the answer. If you get them wrong, these may highlight a gap in understanding the concepts. If you get the answer wrong or you do not know the answer, please click on the "Do not know" option to direct you to the learning materials which may explain the concepts in detail. Click on the Next link in SECTION B to go through to more challenging questions.

SECTION C provides a more advanced question on the same topic for you to apply your knowledge on the subject matter. You will need to run your answer to this question in any IDE (Integrated Development Environment) such as BlueJ, Eclipse or in command-line.

Figure 1 explains the logic behind this tool represented in terms of the Bloom's Taxonomy levels of "Remember", "Understand" and "Apply".

Go to main page
In each of the webpage for the exercises, we include brief instructions on how to use the guided learning tool. However, novices may also revisit the introduction page using the instruction link newly inserted.

### 8.3.2 Example on Each Topic

In guided learning A, there are two sections of questions: debugging and multiple choice questions. We rated the debugging questions as the higher level in Bloom’s Taxonomy when compared to the multiple choice questions. More than 70% of the novices who trialled the guided learning A managed to fix and detect four of the errors. But fewer than 20% of the novices who trial the guided learning A managed to detect e3. In order to detect and fix e3, novices have to be able to recall (a skill in remember level of Bloom’s Taxonomy) that the Java language is case sensitive. Plus, some novices were unable to answer Question 2 in Section B of guided learning A correctly. This question is classified in the understand level of Bloom’s Taxonomy. Therefore, in guided learning B, we use multiple choice questions to focus on exercises for novices to remember and understand programming code terminologies and fundamental concepts.

We also include an example of a program for each topic as recommended by the novices. The example program is placed in Section A of guided learning B, where the hyperlinks have been included to link the programming terms to the questions in Section B.

In guided learning B we still classify the questions based on Bloom’s Taxonomy. We create Section C with advanced exercises, building upon Section A and Section B, whereby novices may apply their knowledge to write a program.

In Section B there exist multiple levels of difficulty in the multiple choice questions in the remember and understand levels of Bloom’s Taxonomy. Therefore, we use our instructor levels of complexity which include syntax knowledge, semantic knowledge, problem solving skills and level of difficulty, to arrange the multiple choice questions in increasing levels of complexity (refer Section 7.3.3). We also increase the number questions in Section B of the guided learning B and include more topics with further questions as requested by novices.

### 8.3.3 Webpage Design

In guided learning A, the two sections A and B are on the same webpage. Novices recommended improvements to the layout. Consequently, we create guided learning B with four separate windows on the webpage. A window for each of the Menu, Section A, Section B and
Section C of the *guided learning B*. We develop the windows using frames, so that the novices are free to navigate in Section B while the other sections maintain their current locations.

A sample screenshot of the layout in *guided learning B* is in Figure 8.5 and other screenshots are in Appendix F.

### 8.4 Cognitive Process Representation Framework

In our reflections to *guided learning A*, we attempt to present a framework to relate the task and exercises with levels of the cognitive processes in Bloom’s Taxonomy. Based on the data collected from *guided learning A*, we present four novices’ responses in Table 8.6 where $Q1$ denotes Question 1, $Q2$ denotes Question 2 and so on. The correct answer will be represented in solid green arrow and the incorrect answer will be represented in dashed red arrows on the diagram in Table 8.6.

We classify Question 1 in the *remember* level and Question 2 in the *understand* level of Bloom’s Taxonomy. Novice3 answered Question 1 correctly but was not able to answer Question 2 correctly. Novice8 did not recall or *remember* the terminology *io* and therefore this novice was not able to *understand* the usage of the import statement in Question 2.

Novice6 did not answer either Question 4 or Question 6 correctly. There is no connection between these two components, which can be seen by the red dashes in different places on the diagram. Novice7 managed to choose the correct answer for all questions represented by solid green arrows.

Based on our reflections of novice responses to the multiple choice questions in *guided learning A*, we introduce the representation of cognitive processes to learn computer programming for novices in *guided learning B*. This is to aid novices and instructors monitor novices progress in learning to program. This proposed framework is included on the introduction page of *guided learning B* to explain to novices how the tasks are related to building their mental models of programming.

We also introduce further exercises in the *apply* level of Bloom’s Taxonomy in *guided learning B* as we are interested to find out whether novices who correctly answer all the multiple choice questions are able to *apply* their knowledge in writing a complete program. If this is true, these novices will portray the *relational* level of the SOLO Taxonomy, whereby they can relate all the components introduced and are able to apply their knowledge to write computer programs.

In the next section we discuss novice responses to *guided learning B*. 

174
Hello Shuhaida, Date: Sun, 08 Jan 2012 03:23:13

Learning objective: In this topic, you are exposed to operators and data types.

Instructions: Click on the hyperlinks in SECTION A for more questions in SECTION B to aid you to learn about the particular subtopic.

## SECTION A

The following program print out:

```
ADDITION: x + y = 50
SUBTRACTION: x - y = 10
MULTIPLICATION: x * y = 600
DIVISION: x / y = 1
```

```java
public static void main(String[] args) {
    int x = 30;
    int y = 20;
    int z;
    
    z = x + y;
    z = x - y;
    z = x * y;
    z = x / y;
    
    System.out.println("ADDITION: x + y = " + z);
    System.out.println("SUBTRACTION: x - y = " + z);
    System.out.println("MULTIPLICATION: x * y = " + z);
    System.out.println("DIVISION: x / y = " + z);
}
```

## SECTION B: Select the answer from the options given.

What will be the value assigned to the variable `x` as a result of the following statement?

```
int x = 10 / 4 + 10 % 4 * 2;
```

- A. 0
- B. 4
- C. 6
- D. 8
- E. None of the above

## SECTION C: Advance exercise for you to try:

Improve the programming code in SECTION A by creating a menu for users to choose on the operation, insert the digits and calculate it and give the result in two decimal points.
Table 8.6: Guided Learning A: Visual Representation of Individual Novice Cognitive Processes Based on Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Respondent</th>
<th>Representation of Cognitive Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novice3</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Novice8</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Novice6</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Novice7</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
8.5 Responses to Guided Learning B

We received 18 responses from the JPG2 cohort. There were 32 novices enrolled for JP, representing 56.25% of the class. Novices were attracted to try the multiple choice questions in Section B in the guided learning B. None attempted to answer Section C, the question classified in the apply level of Bloom’s Taxonomy.

Based on our observations none of the novice answered the questions in order. Instead they answered the questions apparently randomly, possibly because the prioritised particular areas of greatest interest.

Another interesting fact was that of the seven novices who attempted Question 2, only four novices managed to find the correct answer, option A, on their first attempt. All the others selected option B on their first attempt and when they realised that option B was the incorrect answer, they changed their answer to the correct answer, option A. For the same question asked in guided learning A we received six correct responses with the other five incorrect.

We present the cognitive process representation framework for four random responses in guided learning B (refer Table 8.7). Green arrow represents the correct answer and the red-dotted arrow represents that the novice chose a distractor. The number on the arrow head is an indication of the number of attempts to answer the question, if it is more than once. Novice1 fail to choose the correct answer for Question 2 and on the second attempt, Novice1 managed to choose the correct answer. Novice2 only chose to answer Question 1, Question 2 and Question 7 and got the correct answer for the three questions. Novice3 managed to choose the correct answer for the three question attempted. Novice1, Novice2 and Novice3 only attempted the first set of multiple choice questions in guided learning B, Module 1.

Novice4 chose the wrong answer for the first attempt, but managed to get the correct answer on the second attempt for Question 2. Novice4 attempted two questions from the second modules and got both correct. Based on the arrows both questions 12 and 20 are in the understand level of Bloom’s Taxonomy, and we classify them based on the instructor levels of complexity as explained in Section 7.3.3.

Novices were encourage to provide their feedback to the guided learning tool. For guided learning B, we receive only one suggestion from the novices and it was to provide more examples. These will be included in our future work which will provide a bank of multiple choice questions that have different levels of difficulties. On top of this suggestion, in the next section we provide the data collected from the survey about the guided learning tool,
Table 8.7: Guided Learning B: Visual Representation of Individual Novice Cognitive Processes Based on Bloom’s Taxonomy

<table>
<thead>
<tr>
<th>Respondent</th>
<th>Representation of Cognitive Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novice1</td>
<td>![Diagram for Novice1]</td>
</tr>
<tr>
<td>Novice2</td>
<td>![Diagram for Novice2]</td>
</tr>
<tr>
<td>Novice3</td>
<td>![Diagram for Novice3]</td>
</tr>
<tr>
<td>Novice4</td>
<td>![Diagram for Novice4]</td>
</tr>
</tbody>
</table>
including guided learning A and guided learning B.

8.6 Survey Responses

In this section we highlight the responses by novices about the guided learning tool. Their responses appear in Table 8.8. P1G1 responded to the survey for guided learning A and JPG2 responded to the survey for guided learning B. We asked the same questions in the survey as we are evaluating on the same basis of the guided learning tool, that is the learning tasks or exercises for both versions of the guided learning tool are scaffolded using Bloom’s Taxonomy.

In the second evaluation with the JPG2 for guided learning B, we shortened the Likert scale to only provide options Disagree, Neutral and Agree in the survey. This was because in the same survey of the P1G2 group, we did not receive many responses in the two groups (strongly disagree and strongly agree). Hence they were not included in the second survey. There were 29 novices in total from the P1G2 and JPG2 groups. We present novice responses in Table 8.8. Note that the number in brackets for Table 8.8 and Table 8.9 represents the actual number of responses received.

We received positive feedback from the novices to the statement saying that they felt these exercises can help them to learn programming, a claim supported by two novices strongly agreeing (from the P1G2 group) and 15 novices agreeing (from P1G2 and JPG2 groups) with the proposition. Nine novices agreed that these exercises amounted to extra work for them, whereas eight disagreed with them being extra work, so this was borderline but possibly explains the motivation of the novices to use the guided learning B. Sixteen novices disagreed with the statement that the exercises were too hard and only 6 agreed with the statement that the exercises were too easy. We received 12 neutral responses for the statement that the exercises were too hard and 18 neutral responses for the statement saying they were too easy. We interpret this as approximately half of the novices who trialled the guided learning tool considered the questions to be neither too easy nor too hard. Two strongly agreed and 17 agreed with the statement that the exercises helped them to understand the basic structure of a program, which was precisely the learning objective of the exercises.

Furthermore, 17 novices agreed with the statement that the exercises helped them to recall the required information in learning programming. Fourteen novices agreed with the statement that the exercises helped them with their misunderstandings about programming concepts.
### Table 8.8: Responses to Survey (Part 1) from P1G2 and JPG2 groups

<table>
<thead>
<tr>
<th>Questions</th>
<th>Group ID</th>
<th>Number of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. These exercises help me learn programming</td>
<td>P1G2</td>
<td>0 6.9% 3.4% 20.7% 6.9%</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>6.9% 24.1% 31.0%</td>
</tr>
<tr>
<td>b. They are simply extra work for me</td>
<td>P1G2</td>
<td>0 6.9% 10.3% 20.7% 0</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>20.7% 31.0% 10.3%</td>
</tr>
<tr>
<td>c. They are too hard</td>
<td>P1G2</td>
<td>0 17.2% 17.2% 3.4% 0</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>37.9% 24.1% 0</td>
</tr>
<tr>
<td>d. They help me to understand a basic structure of a program</td>
<td>P1G2</td>
<td>0 3.4% 3.4% 24.1% 6.9%</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>6.9% 17.2% 34.5%</td>
</tr>
<tr>
<td>e. They could help with any misunderstandings</td>
<td>P1G2</td>
<td>0 3.4% 6.9% 27.6% 0</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>13.8% 27.6% 20.7%</td>
</tr>
<tr>
<td>f. They are too easy</td>
<td>P1G2</td>
<td>0 6.9% 20.7% 10.3% 0</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>10.3% 41.4% 10.3%</td>
</tr>
<tr>
<td>g. They help me to recall the required information</td>
<td>P1G2</td>
<td>0 3.4% 6.9% 27.6% 0</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>3.4% 27.6% 31.0%</td>
</tr>
<tr>
<td>h. I would like extra exercises like these to help me learn programming</td>
<td>P1G2</td>
<td>3.4% 3.4% 17.2% 10.3%</td>
</tr>
<tr>
<td></td>
<td>JPG2</td>
<td>6.9% 17.2% 34.5%</td>
</tr>
</tbody>
</table>
We also received positive feedback (three *strongly agreed* and 15 *agreed*) that they would like the guided learning tool to help them as they continue to learn programming. Overall, more than half of the novices considered the guided learning tool helpful to aid them in learning programming.

Note that for Question d and Question h (Table 8.8) one novice did not respond to each question and this resulted in 28 responses for each of the questions. However, we would like to state that it was not the same novice who did not answer both questions.

Table 8.9: Responses to Survey (Part 2) from P1G2 and JPG2 groups

<table>
<thead>
<tr>
<th>Questions</th>
<th>GroupID</th>
<th>Number of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Yes, Yes, Not</td>
</tr>
<tr>
<td>i. I accessed the Internet to help me answer</td>
<td>P1G2</td>
<td>0 0 37.9% (11)</td>
</tr>
<tr>
<td>the questions</td>
<td>JPG2</td>
<td>27.6% (8) 10.3% (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24.1% (7)</td>
</tr>
<tr>
<td>j. I used the course materials to help me</td>
<td>P1G2</td>
<td>3.4% (1) 3.4% (1)</td>
</tr>
<tr>
<td>answer the questions</td>
<td>JPG2</td>
<td>17.2% (5) 17.2% (5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27.6% (8)</td>
</tr>
<tr>
<td>k. I used a text book to help me answer the</td>
<td>P1G2</td>
<td>0 0 37.9% (11)</td>
</tr>
<tr>
<td>questions</td>
<td>JPG2</td>
<td>13.8% (4) 10.3% (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37.9% (11)</td>
</tr>
</tbody>
</table>

We further investigate the references that the novices use to help them to answer the questions in the guided learning tool. Novices may choose between *yes, a lot*, *yes, a little* and *not at all* as answers for this question. The *yes, a lot* response refers to novices frequently referring to the resources listed in the survey to aid them in answering the questions. The *yes, a little* response refers to novices rarely referring to the resources and *not at all* means novices did not refer to any of the resources to guide them in answering the questions, in the guided learning tool.

Our results in Table 8.9 show that novices from the P1G2 group did not use help from the Internet or textbook at all. However eight novices from the JPG2 used the Internet *a lot* while three used it *a little* and seven did not use it at all when answering the questions in the guided learning tool. From P1G2, one novice used the course material *a lot* and one novice used it *a little* to help them answer the questions. From JPG2, five novices referred *a lot* and five referred *a little* to the course material. From JPG2 four referred to textbook *a lot* while three referred it *a little* to help them answer the questions in the guided learning.
Hence we conclude that, for the P1G2 group, novices did not refer to the resources from the Internet and textbook, and relied on the current state of their cognitive models and some refer to the course materials to answer the questions in the guided learning tool. While more novices from the JPG2 group seemed to refer to other resources a lot to help them in answering the questions in the guided learning tool, it was still not the majority of students. This may be because JPG2 cohort is more experience and they may know to source material outside the course. Therefore in future work we may include more useful links in the guided learning tool as additional learning resources for novices.

8.7 Novice Performances in Programming Assessment

Programming 1 is a critically important core unit for all the programs in our range of computer science degrees. It is a prerequisite for Programming 2 and Java for Programmers. These two courses are prerequisites for Computing Theory and Programming Techniques. Therefore, if novices fail Programming 1, it will be difficult for them to progress on to the next course.

In order for the novices to pass Programming 1, they must pass both the formative and summative assessments. The distribution of weights that contribute to the final grade of Programming 1 is shown in Table 8.10.

The final grading is the total of the formative and summative assessments. The formative assessments contribute to a total of 60% of the final grading and consist of: 30% for assignments, 10% for weblearn tests, 10% for a mid-semester test and one percent mark awarded for completing the tasks required in the tutorial sessions. The remaining 40% is from the final exam summative assessment.

The total marks each novice receives will determine the grades they achieve. There are five grades possible if the students attend the classes and pass both formative and summative assessment. High Distinction (HD) is the highest grade and Fail (NN) is the lowest grade that a student may achieve.

No grades are awarded if students withdraw (WDR) from the course before the designated deadline, usually a month from the commencement date of the course, whereas Did Not Sit (DNS) means the a student did not attend the final examination, yet was still enrolled in the course.

In Section 8.6 novices evaluated and provided their opinions about the guided learning tool.
and guided learning B. Overall, we seek to determine whether their learning had improved through using the guided learning tool. We analyse the performances of novices in terms of their marks that contributed to the overall assessments.

We studied the assessment results for the two cohorts, P1G2 and JPG2, and compared them with the results of novice cohorts from the previous year, namely, P1G1 and JPG1. Table 8.11 presents a list of final grades attained, shown in Figure 8.6 for P1G1 and P1G2 groups, and also Figure 8.7 for JPG1 and JPG2 groups.

We present novice performances in Figure 8.6 and Figure 8.7. Also, the grade distributions for the various cohorts are shown in Table 8.12.

Figure 8.6 compares the results in the summative assessments for Programming 1 for Semester 1, 2009, and Semester 1, 2010. The number of novices who scored “high distinction” (HD) increased over the year. Also, the number of novices who failed, increased. On the other hand, for P1G2, the number of novices who withdrew from the course decreased, and
all the novices who enrolled for the course attended the examination.

JPG1 and JPG2 had fewer novices enrolled compared to P1G1 and P1G2. More than half the class tried and evaluated the guided learning tool. Referring to Figure 8.7, there was an increase in the number of novices who obtained “high distinction” in JPG2 as compared to JPG1. Also, the number of novices who failed decreased in JPG2.

Although there were improvements in novice performances as evidenced by the results, as well as improvements to the retention rate in the two groups who trialled the guided learning tools, we cannot conclude that this is a direct result of using the guided learning tools. However, we posit that the guided learning tools may have contributed to such success.
Overall, we received positive feedback from novices about the guided learning tool. Novice programmers are confronted with too much information and too many skills to grasp in their first few weeks of a semester.

As an example, in an evaluation of guided learning A, only two novices managed to discover that Java is case sensitive and that the program terminates with an unhelpful error message, if this invalid case is used. Novices may have overlooked this but we emphasise that the exercises in the guided learning tool were to show how important it is that they be made aware of the entire set of basic rules, so that they do not continue to make simple mistakes.

Almost half of the novice responses (from guided learning A and guided learning B) suggested that the purpose of “import java.io.*;” is to import the input and output of the program to other systems, which means they were unable to understand the use of the import statement. We highlight this question to illustrate that what some assume as basic and simple may be complex and misunderstood by others. Moreover, such misunderstandings may be compounded and result in a significant negative impact if they are not addressed and corrected in the early stages of learning.

Most of the novice programmers scored better in the multiple choice questions section than in the debugging question section. This is because the set of multiple choice questions was associated with lower levels of Bloom’s Taxonomy.
CHAPTER 8. EVALUATION OF GUIDED LEARNING TOOL

We hope that by guiding the novices through test questions that cover multiple choice questions, which in turn cover multiple cognitive levels of Bloom’s Taxonomy, they will be able to construct suitable mental models of cognitive processes required to learn programming. Additionally, in each cognitive level of Bloom’s Taxonomy, we categorised the questions based on our proposed instructor levels of complexity measure, which contains four criteria, including syntax knowledge, semantic knowledge, problem solving skills and level of difficulty.

When we surveyed the novices about the guided learning tool, almost half of the responses disagreed with the statement that the questions were too hard and approximately half gave a neutral response to the statement that the questions were too easy. We infer from the previous statement that, in the first few weeks of learning programming, while novices do not find learning programming too difficult, they would not say that it is too easy either. Therefore, during this critical learning period (the first four weeks of a semester) novices should be steered towards suitable questions to help them to learn and avoid misconceptions that may propagate further misunderstandings in later weeks.

In the first week the guided learning tool may help novices to remember and understand a program and help them to master basic syntax of a program. Furthermore, as the weeks progress novices should be able to strengthen their syntax and logic skills by practising the multiple choice questions on assignment, operators, selection and iteration topics (these exercises are included in guided learning B).

Novices were also concerned about the lack of instructions in guided learning B. Novices were not familiar with the environment and the learning approach was new to novices. As there were a few windows containing multiple levels of programming exercises, they did not know which one to start with. Therefore, we included an introduction page that explained in detail how the sections were interrelated. We also introduced novices to Bloom’s Taxonomy to show them how we classify the questions. We hope this explain the layout of the questions that have multiple levels of difficulties.

We explained our proposed framework in the introduction page of guided learning B. Instructors may use this framework to probe the minds of novices in their self-navigating exploration, to learn to program. Novices and instructors may use this framework to identify novices’ weaknesses and strengths. Instructors may improve their teaching if they would know “which” programming concepts they should emphasise to help novices overcome their learning difficulties. Early misconceptions can be resolved if the learning barriers can be identified. Having a strong fundamental knowledge of programming concepts may help novices progress easier to advanced programming courses.
Finally, we were pleased to see that novice performances improved for the P1G2 and JPG2 groups (who trialled the guided learning tools) compared to P1G1 and JPG1 (the control group). As far as we are aware, the course materials and the delivery of the course remained the same for P1G1 and P1G2 and correspondingly, for JPG1 and JPG2. There may be other factors involved in this success, but we believe that the guided learning tool may be one of the factors contributing towards the novices’ performance improvement.

8.9 Summary

We trialled the guided learning tools with two cohorts of novice programmers. Based on the surveys, novices gave us positive responses about the guided learning tools. We also received suggestions of how to improve the guided learning tools in terms of adding more questions, examples and a more comprehensive layout for the interface as compared to the original layout in the guided learning tool.

In this chapter we highlighted not only the evaluation of both guided learning A and guided learning B approaches and tools, but we introduced a framework for cognitive processes representation as a method to explore the novices’ learning processes about programming. We have provided a few examples of the cognitive processes representation based on novices’ responses to guided learning A and guided learning B. We discuss this framework in the introduction page of the guided learning B so instructors and novices can use this framework to monitor and improve the novices’ learning of programming.
Chapter 9

Conclusions

In this thesis we have been concerned with researching the cognitive models of programming concepts imagined by novice programmers. We would like to be able to probe their minds to see the models they have invented of elementary programming concepts and identify invalid models early in their learning, to avoid problems of understanding further on in their studies. To this end we have developed a tool and a guided learning approach to both identify and explain cognitive models in the minds of novices.

We surveyed existing introductory programming tools and emphasised that there is a need for better instructions and organised course content, to scaffold the fundamental learning of programming. We studied existing learning theories which we have modified, revised and applied to a learning approach for novice programmers.

We also have extensively studied the issues surrounding multiple choice questions in programming assessments, including the actual responses of novices in summative assessments and the programming instructor perspectives. We benefited from the use of multiple choice questions as the basis of our proposed learning tool. We have learned that multiple choice questions can be a powerful test instrument. If prompt answers are provided they can be used to motivate novices to try again if they chose an incorrect answer.

As a result of our investigation, we proposed the guided learning tool based on Bloom’s Taxonomy, which helps novices to scaffold their learning of programming. The initial version is labelled \textit{guided learning A} and the improved version is labelled \textit{guided learning B}. \textit{Guided learning A} has two sections of questions: debugging questions and multiple choice questions. \textit{Guided learning B} has three sections: the task, multiple choice questions and writing a complete program.
In this chapter we summarise our overall findings for the data collected through surveys, self analyses and evaluations by novices of the guided learning tool. Based on the data collected, we present our conclusions and discuss possible directions for future research. We revisit our research questions from Chapter 1 to explain and validate our research.

9.1 Research Question 1: How might one classify multiple choice questions in summative assessment for novice programming?

Initially, we found that introductory programming courses are much more difficult than realised and have an inadequate classification of difficulty for assessments. Usually, the questions for summative assessments are selected based on suitability and coverage of topics and not based on the levels of difficulty and the variations of skills required to answer them.

We researched the original and revised Bloom’s Taxonomy, as well as the SOLO Taxonomy, which resulted in our redefinition of the scale in Bloom’s Taxonomy to suit the difficulty levels involved in fundamental programming courses. In addition, we defined a measure of complexity to explain instructor levels of expectation of the skills required to answer each question created. We defined a separate measure to explain the difficulty faced by novices when answering these questions in summative assessment.

We developed classifications that combine the original and revised Bloom’s Taxonomy, the SOLO Taxonomy, the instructors levels of complexity and the novices levels of difficulty to adequately explain the classification of difficulty in multiple choice questions tested in programming summative assessments. In our analysis we found that these three classifications proposed (Bloom’s Taxonomy and instructor levels of complexity and the novice levels of difficulty) are not related to each other. There exists a moderate correlation between novice levels of difficulty and the index of discrimination.

9.2 Research Question 2: What are the programming instructors’ views of multiple choice questions in summative assessment?

Conflict between novices or students and instructors may be one of the reasons that cause learning difficulties in programming courses. Instructors may assume a fundamental course is easy, but actually some novices may struggle to grasp the details of learning programming very early on.

We surveyed instructors about multiple choice questions and we found that almost half of the instructors who responded believed that summative assessment is, and is meant to
be a valid measure of a student’s ability to program. This is despite programming being a very practical skill, practised in an environment which gives errors when the code does not work, but no feedback about the quality of the code when it does work. Instructors would not expect a novice who could program to fail a summative assessment test nor would they expect a person who could not program to pass the test. There were 27.

Other than that, 50% of the instructors from the survey agreed and strongly agreed with the statement that multiple choice questions are suitable to test low levels of understanding and think these are easy questions.

Furthermore, a total of 47% of the responses were completely confident and confident that multiple choice questions test novices’ understanding of programming concepts. Therefore, using the multiple choice questions may benefit the learning tool proposed to test the understanding of fundamental programming.

We also received recommendations that essay format should be included to test novice ability to write programs. Some instructors portray multiple choice questions as being easy summative assessments for programming courses, used to keep the weak students on track, to test students’ basic knowledge, to keep the students happy, and to keep them motivated, but some also equated them to distributing free marks. While we accept that some instructors resist using multiple choice questions because they are seen as easy questions, we have shown in earlier chapters that not all multiple choice questions were found to be easy by the students.

Instructors agreed and strongly agreed that they devise a question in summative assessment based on coverage of particular topics with certain levels of difficulties. In addition, more than a third of the instructors surveyed reported their confidence in using multiple choice questions in assessment, on the condition that they are good quality multiple choice questions. Therefore summative assessments should not only cover particular programming topics, but instructors must also consider the levels of knowledge and skills on which they are testing the students.

In order to create a high standard of multiple choice questions, we established the criteria for our instructor levels of complexity. We aimed to explain the instructor approaches designing multiple choice questions. We introduced four criteria: syntax knowledge, semantic knowledge, problem solving skill and level of difficulty. We applied these criteria to selected programming questions. We asked the instructors to apply them to the questions as well so we could identify their perspectives of the questions tested in summative assessment.

Based our analysis, there existed significant correlations between the four criteria introduced (syntax knowledge, semantic knowledge, problem solving skills and level of difficulty).
If one criterion is rated at the medium level of difficulty, say for syntax knowledge, it is most likely that another criterion will also be rated at a medium level of difficulty, such as problem solving skill. However, this does not hold true in all cases.

We also found that experience is not a factor that can differentiate how instructors evaluate programming questions. There exists no significant correlations between semesters of teaching and semesters of devising exam questions and the responses that instructors gave about the levels of syntax knowledge, semantic knowledge, problem solving skills and level of difficulty tested in summative assessment. This debunks some popular but incorrect perceptions held by novices about how more experienced instructors evaluate questions harder compared to less experienced instructors. The difficulty levels of question were not related to the instructors’ experiences.

As a whole, from the responses to the instructor survey, and from the literature surveyed, multiple choice questions can provide a reliable test for the Programming 1 course. Therefore, we decided we wanted novices to benefit from using multiple choice questions in our proposed learning tool.

9.3 Research Question 3: How can a guided learning tool aid novices in learning programming?

We proposed the guided learning tool that provides questions with different levels of difficulties for novices. The guided learning tool was developed to cater for different individual needs in learning to program. Different individuals have different levels of capacities and capabilities to learn. Each individual learns at different pace, and therefore, it is not easy to develop a learning aid to suit every novice. In developing the idea of the guided learning approach and tool, we accept that the human cognitive processes are unique and that learning programming is difficult, but not for all novices.

We developed the guided learning tool in two cycles to carefully test our idea of the proposed learning approach. Guided learning A was trialled by a group of novices and their suggestions as well as our own reflections directed us to develop the improved version of the guided learning tool, which we labelled as guided learning B.

There are three sections in guided learning B; the task, multiple choice questions and writing a complete program. In Section A, novices may learn through reading the sample program. In the sample program, there are hyperlinks linking to the questions in Section B. Novices may self-navigate through Section B to explore their learning needs. In terms
of the amount of information to be processed, the guided learning tool is flexible because novices may try any number of questions, any number of times. Novices may also answer the questions in any level of Bloom’s Taxonomy, not necessarily beginning with the lowest level, and working their way up to the higher levels. Therefore, the guided learning tool also supports top-down as well as the bottom-up learning approach [Sun and Zhang, 2004]. Bloom’s Taxonomy may aid both deep and surface learning approaches [Ramsden, 1992] as novices may explore the exercises based on their curiosity.

For novices who find learning programming to be difficult and often obtain low grades in the programming courses, it is not easy to identify their learning difficulties. As discussed in earlier chapters, we studied the introductory programming courses’ assessments in order to understand the novice learning difficulties. Hence, we proposed our guided learning tool to probe the minds of novices to monitor their progress and, most importantly, to identify the weak components of their mental models that contribute to their misunderstandings or misconceptions in learning programming. We hope that the guided learning tool can identify and help the instructors and novices to fix early misunderstandings or misconceptions of fundamental programming concepts.

In Figure 9.1 we present our framework for representation of the cognitive processes. Our aim is for novices to construct suitable mental models to program that can be achieved by building the cognitive processes using our proposed framework of cognitive processes representation, as shown in Figure 9.1. In a fundamental level programming course, we emphasise only levels up to the apply level of Bloom’s Taxonomy, whereby at the end of the four weeks of content using the guided learning tool, novices are able to write a working computer program.

If novices are able to achieve the cognitive processes as portrayed in Figure 9.1, novices have achieve the relational level in the SOLO Taxonomy (refer to Figure 9.2), whereby multiple components of relevant knowledge are related.

These components may consist of the topics in detail for learning to program, including the terminologies, symbols, concepts, theories and practical tasks. As the guided learning tool is intended for the first four weeks of a fundamental programming course, we would not expect novices to achieve the extended abstract level of the SOLO Taxonomy, whereby novices may generalise their set of knowledge to a new domain.

We posit that the cognitive processes representation framework for guided learning tool may support the constructivist theory of learning. Using the different levels of Bloom’s Taxonomy, the guided learning tool supports the constructivist theory of building mental
CHAPTER 9. CONCLUSIONS

Figure 9.1: Cognitive Processes Representation When Learning To Program Using Guided Learning Approach and Tool

Figure 9.2: Relational Level of SOLO Taxonomy
models, that aims to make sense of the program as a whole (not line by line) and enable novices to relate each component to another within a program.

We have developed and evaluated the guided learning tool, which focuses on helping novices to learn to program based on the first four weeks of learning contents for an introductory programming course. The exercises in the guided learning tool are scaffolded using the Bloom’s Taxonomy. Novices may try the multiple classifications of exercises to strengthen their foundations in programming, especially with the syntax knowledge, which has limited the capabilities of novices to learn programming in the early weeks of the course. We found that novices were generally happy to trial the tool and many stated that the guided learning tool could have been useful for them in their early stages of learning programming. In the surveys, approximately two thirds of the responses agreed or strongly agreed that novices would like to use the exercises in guided learning tool to help them learn programming. Additionally, novices suggested that they want more exercises and examples for all modules, to be incorporated in the guided learning tool. There may be further work to be done with the interface, but the underlying skills are being practised.

9.4 Conclusion

It is often assumed that basic elementary knowledge is easy to learn and instructors may take it for granted when teaching programming to novices. Most novice programmers in our surveys agreed and strongly agreed that the questions from the first few weeks of the syllabus in a fundamental programming course were easy, but for some novices, their responses to the fundamental concepts of programming in summative assessment did not support such claims.

Hence, we think that it is important to scaffold learning to program to steadily strengthen novices’ knowledge and to help them progress to the more difficult learning practises in programming, based on a strong foundation. We aim to probe the minds of novices using a guided learning tool to determine their state of knowledge, about the fundamental concepts.

We tested novice programmers on basic knowledge during their early weeks of learning programming and proved that while some students were able to detect and fix basic programming errors and understand the basic jargon and its usage, others could not. Our aim is to aid novices in learning programming by practising exercises and learning from them. We classified the multiple choice questions using Bloom’s Taxonomy and then arranged the questions in each level based on our instructor levels of complexity (syntax knowledge, se-
mantic knowledge, problem solving skill and level of difficulty) to help novices decompose learning problems in programming. We received positive feedback that approximately two thirds of the novices agreed or strongly agreed that the exercises in the guided learning tool helped them to learn programming.

9.5 Future Works

There were 21% of the instructors we surveyed who disagreed or strongly disagreed with using summative assessment as a measure of a student’s ability to program. In future work, we would like to investigate this further, with a view to identifying other options for assessing a student’s ability to program. For the present, based on our survey, when almost half of the instructors are happy to use summative assessment, we have settled for encouraging the students to practise typical exam questions and in particular multiple choice questions.

Hence, we plan to establish the use of our proposed criteria for instructor levels of complexity which include syntax knowledge, semantic knowledge, problem solving skill and level of difficulty to add variations to the different levels of questions used to construct novices’ mental models of programming. Our research has highlighted that these four criteria are not discussed in depth amongst instructors and often not applied when writing summative programming assessment.

With regards to the guided learning tool, we will add a link for the option of “I do not know” in Section B of the guided learning B to related resources, such as the course module materials or other reliable sources from the Internet or books, to aid novices to answer the questions. Based on our survey, novices do not seem likely to refer to these resources much when they trialled the guided learning tool. Learning programming requires one to explore other sources of knowledge and sometimes referring to the course materials only are not enough.

We also plan to add more questions to the guided learning tool. We believe that the more a novice uses the guided learning tool, the clearer the mapping of the cognitive processes should be for the instructor. The questions and responses analysis section of our guided learning tool may show the question number and the number of attempts made to answer the question. This will benefit instructors and novices to know how many attempts have been made and how much the novices have learned. The framework of cognitive processes representation may also detect the novices’ learning approaches to allow for strategies to be developed that support personalised learning, in order to conquer programming courses.
Finally, we would like to expand the cognitive processes representation to all the levels in Bloom’s Taxonomy. The expansion to all the levels of Bloom’s Taxonomy requires further analysis because, in the apply level and upwards, it may require combinations of components learnt. Thus, the framework may be used not only for novices, but it could also probe all level of learners’ cognitive processes.

9.6 Concluding Remarks

Our main research contribution has been to address and improve the learning difficulties in programming during the first few weeks, using the guided learning tool. We probe the minds of novices and apply a framework which maps their progress in the guided learning tool. The aim has been to help novices advance to the subsequent programming experiences more smoothly. We have identified keywords and some levels for scaffolding but this can be expanded at any stage. We do not plan to change the existing course content, but to add to our guided learning tool, which contains programming exercises gathered from the learning contents from the early part of the teaching period (weeks one to four) to overcome the cognitive (process of thinking) gap faced by novices in the early stages of learning to program. We have researched Bloom’s Taxonomy, instructor levels of complexity and novice levels of difficulty as measures to classify multiple choice questions in assessments. We have proposed our own measures of levels of complexity and difficulty. We also researched how the SOLO Taxonomy may be applied to measure cognitive processes for short answer questions in programming summative assessment.

Overall, we have essentially developed an empirical driven synthesis of theoritical frameworks, incorporating the student and instructor perspectives, related to improving the early novice programmer learning and assessment process. These provide a framework by which educators can improve the rigour in the assessment activities to design more consistent, equitable and appropriate multiple choice question assessments for early novice programming students, apply these to formative and summative assessments, and hopefully thereby increase student chances of early success and persistence in their courses. The ability for students to test themselves online to their own schedule in early stages of the course, without fear of criticism or rebuke, is also an important rationale for the use of self assessment tool. Accordingly this synthesis of framework has been tested with promising results in initial studies using a prototype of the Guided Learning Tool in two cycles of Action Research.
Appendix A

Ethics Approval
25th February 2008

Mohamed Shuhidan
K4 High Street
Windsor VIC 3181

Dear Mohamed

SETAPP 80 – 07 SHUHIDAN A constructivist approach to learning programming

Thank you for submitting your amended application for review.

I am pleased to inform you that the committee has approved your application for a period of 3 Years to February 2011 and your research may now proceed.

The committee would like to remind you that:

All data should be stored on University Network systems. These systems provide high levels of manageable security and data integrity, can provide secure remote access, are backed up on a regular basis and can provide Disaster Recover processes should a large scale incident occur. The use of portable devices such as CDs and memory sticks is valid for archiving, data transport where necessary and for some works in progress; The authoritative copy of all current data should reside on appropriate network systems; and the Principal Investigator is responsible for the retention and storage of the original data pertaining to the project for a minimum period of five years.

Annual reports are due during December for all research projects that have been approved by the Human Research Ethics Sub-Committee.

The necessary form can be found at:
http://www.rmit.edu.au/rd/hrec

Yours faithfully,

Barbara Polus
Chair, Science Engineering & Technology Portfolio
Human Research Ethics Sub-committee ‘B’

Cc HRE-SC Member: Diana Donohue Chair of HRE-SC ‘A’
Supervisors: Margaret Hamilton School of Computer Science & IT
Daryl D’Souza School of Computer Science & IT
Appendix B

Biggs’ Survey
Invitation to Participate in a Research Project

Project Information Statement

**Project Title:** Debugging: A constructivist approach to learning programming

**Investigators:**
- Miss Shuhaida Mohamed Shuhidan, (PhD student, School of CS&IT, RMIT University)
  s3181759@student.rmit.edu.au, (03) 99252758
- Dr Margaret Hamilton, (Senior Lecturer, School of CS&IT, RMIT University)
  margaret.hamilton@rmit.edu.au, (03) 99252939
- Dr Daryl D’Souza, (Senior Lecturer, School of CS&IT, RMIT University)
  daryl.dsouza@rmit.edu.au, (03) 99252927

Dealing programming student,

You are invited to participate in a research project being conducted by RMIT University. This information sheet describes the project in straightforward language, or plain English. Please read this sheet carefully and be confident that you understand its contents before deciding whether to participate. If you have any questions about the project, please ask one of the investigators above.

Who is involved in this research project? Why is it being conducted?

This research project seeks to understand how people learn to program computers. Any student currently enrolled in Programming 1 or Programming Techniques can be involved in this project.

The project has been approved by the RMIT Human Research Ethics Committee.

Why have you been approached?

We are conducting the research to find out how we can help you learn programming better. We are approaching you as a student enrolled in Programming 1 or Programming Techniques, to be involved as a voluntary participant in this study. We plan to collect data at various stages during the semester, and all data will remain confidential.

What is the project about? What are the questions being addressed?

Programming is a difficult task, but essential for success in Computer Science study. We would like to understand what things make it difficult to learn programming initially, and to see if learning debugging first makes learning programming easier.

If I agree to participate, what will I be required to do?

You will be asked to complete a questionnaire. We also seek consent to access your exam papers with a view to categorizing the type of errors made. This will, in no way affect your mark. For both, the questionnaire and exam papers, you will be providing the student ID; however, it will be de-identified by an independent person to maintain anonymity.

What are the risks or disadvantages associated with participation?

Any participation will remain voluntary and anonymous. Thus, it can be stated that there is no risk to you for being involved in this particular study.

What are the benefits associated with participation?

Your participation will help us to propose ways to improve learning programming. The questionnaires will provide insight into what you might know about programming.

What will happen to the information I provide?
APPENDIX B. BIGGS' SURVEY

We aim to develop a tool to help with learning programming but we do not want to make it more difficult than it already might be, and so we are relying on your advice for us to make informed decisions. Any information you provide will be totally anonymous and will remain confidential. Only the members of the investigative team will see such data, which will be kept secure for a period of five years and then destroyed.

Because of the nature of data collection, we are not asking you to provide written informed consent of participation in the project.

Instead, we assume that you have given consent by clicking the Yes, I agree to participate button below

What are my rights as a participant?

Your rights as a participant of this project include:

- The right to withdraw their participation at any time, without prejudice
- The right to have any unprocessed data withdrawn and destroyed, provided it can be reliably identified, and provided that so doing does not increase any risk for you.
- The right to have any questions answered at any time.

Whom should I contact if I have any questions?

Any member of the investigative team listed at the beginning of this plain language statement may be contacted at any time. Any complaints about the conduct of this research project can be made to the Executive Officer, RMIT Human Research Ethics Committee, see http://www.rmit.edu.au/rd/hrec_complaints and the footer of this page.

Yours Sincerely,

Ms Shuhaida Mohamed Shuhidan, MSc
Dr Margaret Hamilton, PhD
Dr Daryl DSouza, PhD

Any complaints about your participation in this project may be directed to the Executive Officer, RMIT Human Research Ethics Committee, Research & Innovation, RMIT, GPO Box 2476V, Melbourne, 3001.
Details of the complaints procedure are available from the above address or http://www.rmit.edu.au/rd/hrec_complaints
Revised Study Process Questionnaire (R-SPQ-2F)

This questionnaire has a number of questions about your attitudes towards your studies and your usual way of studying.

There is no right way of studying. It depends on what suits your own style and the course you are studying. It is accordingly important that you answer each question as honestly as you can. If you think your answer to a question would depend on the subject being studied, please give the answer that would apply to your course.

You will be asked to fill in the data collection form on the following page. The potential responses to each statement are as follows.

A - this item is never or only rarely true of me
B - this item is sometimes true of me
C - this item is true of me about half the time
D - this item is frequently true of me
E - this item is always or almost always true of me

Please choose the one most appropriate response to each question. Mark the answer sheet in the column that best fits your immediate reaction.

- Do not spend a long time on each item: your first reaction is probably the best one.
- Please answer each item.
- Do not worry about projecting a good image. Your answers are confidential.
- Thank you for your co-operation.

Please enter your student ID: _______________________

Years of Programming Experience: _______________________

Submit Responses...
Revised Study Process Questionnaire (R-SPQ-2F)

Please indicate your agreement with the following statements.

<table>
<thead>
<tr>
<th></th>
<th>A - this item is never or only rarely true of me</th>
<th>B - this item is sometimes true of me</th>
<th>C - this item is true of me about half the time</th>
<th>D - this item is frequently true of me</th>
<th>E - this item is always or almost always true of me</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I find that at times studying gives me a feeling of deep personal satisfaction.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>2. I find that I have to do enough work on a topic so that I can form my own conclusions before I am satisfied.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>3. My aim is to pass the course while doing as little work as possible.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>4. I only study seriously what is given out in class or in the course outlines.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>5. I feel that virtually any topic can be highly interesting once I get into it.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>6. I find most new topics interesting and often spend extra time trying to obtain more information about them.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>7. I do not find my course very interesting so I keep my work to the minimum.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>8. I learn something by rote, going over and over them until I know them by heart even if I do not understand them.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>9. I find that studying academic topics can at times be as exciting as a good novel or movie.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>10. I test myself on important topics until I understand them completely.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>11. I find I can get by in most assessments by memorising key sections rather than trying to understand them.</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Statement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>I generally restrict my study to what is specifically set as I think it is unnecessary to do anything extra.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>I work hard at my studies because I find the material interesting.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>I spend a lot of my free time finding out more about interesting topics which have been discussed in different classes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>I find it is not helpful to study topics in depth. If confuses and wastes time, when all you need is passing acquaintance with topics.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>I believe that lecturers should not expect students to spend significant amounts of time studying material everyone knows won’t be examined.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>I come to most classes with question in mind that I want answering.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>I make a point of looking at most of the suggested readings that go with the lectures.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>I see no point in learning material which is not likely to be in the examination.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>I find the best way to pass examinations is to try to remember answers to likely questions.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix C

Novice Perspectives Survey

This appendix presents the text of selected questions from the exam paper, discussed and analysed in Chapter 4. We have reviewed two sets of examination questions, from semester 2, 2007 and semester 1, 2008. Both sets of examination questions have the same format. The final examinations sheet consist of three sections; Part 1, Part 2 and Part 3.

In Section C.1 we discuss 17 multiple choice questions from Part 1 of the final examination sheets of semester 2, 2007. In Section C.3 we discuss the multiple choice questions and one short answer question which are from Part 1 and Part 2 from the examination sheets of semester 1, 2008. Part 3 involves writing and using classes to produce a complete program and this is not in our main agenda of the research. Part 1 contains 19 multiple choice question, including one fill-in-the blank question. In Part 2, we include the short answer coded question that is analysed using the SOLO Taxonomy.
C.1 Examination Questions for Semester 2, 2007

C.1.1 Part 1

Instruction: For the multiple-choice questions, select the most appropriate response and circle the corresponding letter (A/B/C/D). There is only one correct answer for each question.

1. Which of the following is not a primitive type in Java?
   A) char
   B) String
   C) byte
   D) double

2. What will be the value assigned to the variable x as a result of the following statement?
   int x = 10 / 4 + 10 % 4 * 2;
   A) 0
   B) 4
   C) 6
   D) 8

3. Which one of the following sets of values makes the expression true:
   ( X || !Y ) && ( !X && Z ) ?
   A) X = false, Y = false, Z = false
   B) X = false, Y = false, Z = true
   C) X = true, Y = false, Z = false
   D) X = true, Y = true, Z = true

4. What will be assigned to the variable result after execution of the statements below, if the value that is entered by the user is -20?
   int input = console.nextInt();
   int result = (input == 0 ? 0 : (input < 0 ? -1 : 1));
   A) 0
   B) 1
   C) -1
   D) -20
5. Given that \( x \) and \( y \) are both variables of type int, the statements:

\[
y = x + x; \\
y += y + y;
\]

are equivalent to:

A) \( y = 2 \times x \);  
B) \( y = 4 \times x \);  
C) \( y = 6 \times x \);  
D) \( y = 8 \times x \);

6. What will be the output of the program segment below if marks is input as 40?

```java
System.out.print("Enter marks : ");
int marks = console.nextInt();
if (marks < 50)  
    System.out.print(" Fail");
if (marks < 60)  
    System.out.print(" Pass");
if (marks < 70)  
    System.out.print(" Credit");
if (marks < 80)  
    System.out.print(" Distinction");
else  
    System.out.print(" High-Distinction");
```

A) Fail  
B) Fail Pass Credit Distinction  
C) Fail Pass Credit Distinction High-Distinction  
D) Fail High-Distinction

7. What is the output of the following code segment?

```java
for (int i = 1; i <= 10; i++)
{
    if (i % 5 == 0)
        System.out.print("a");
    else
        System.out.print("b");
}
```

Answer:
8. What is the output of the code segment below?

```java
String prefix = "HelloWorld";
String suffix = ".java";
for (int i = 0; i < 3; i++)
    prefix = prefix + suffix;
System.out.println(prefix);
```

A) HelloWorld  
B) HelloWorld.java  
C) HelloWorld.java.java  
D) HelloWorld.java.java.java

9. How many times will the do-while loop below be executed?

```java
int m = 10;
do {
    System.out.println(m);
    m = m + 1;
} while (m < 10);
```

A) 0 times  
B) 1 time  
C) 10 times  
D) None of the above

10. Which one of the following statements is false?
A) A try block can only have one catch block associated with it  
B) When an exception is caught in a catch block the program automatically goes back to the try block and starts executing any remaining statements after the point where the exception was thrown  
C) Statements in the finally clause will still be executed even if an exception is thrown and not caught  
D) If an exception is not caught in the method where it is first thrown, then the program will immediately terminate with a runtime error
11. What are the values of the variables a, b and c after the execution of the following program:

```java
public class TestParameterPassing {
    public static void main (String [] args) {
        int a = 20;
        double b = 1.5;
        int c;

        c = aMethod(b, a);
    }
    public static int anyMethod (double x, int y) {
        x = x * 2;
        y = y + 10;
        return (int)(x * y);
    }
}
```

A) a = 20, b = 1.5, c = 90  
B) a = 30, b = 3.0, c = 90  
C) a = 20, b = 1.5, c = 30  
D) a = 30, b = 3.0, c = 30

12. Which of the following statements is false?

A) A final class cannot be extended to form a subclass  
B) Methods in a final class cannot be overridden  
C) All methods in a final class must themselves be declared to be final  
D) A superclass cannot be a final class

13. If Account is a class that implements the Printable and Comparable interfaces, then which one of the following statements (assuming any arguments required are being passed correctly) is illegal (will not compile)?

A) Account a = new Account();  
B) Printable p = new Account();  
C) Comparable c = new Account();  
D) All of the statements above are legal.
14. What is the output of the program below?

```java
public static void main (String[] args)
{
    int[][] m = new int[3][3];

    for (int row = 0; row <= 2; row++)
        for (int col = 0; col <= 2; col++)
            m[col][row] = row;

    for (int i=0; i<3; i++)
    {
        for (int j=0; j<3; j++)
            System.out.print(" "+m[i][j]);
        System.out.println();
    }
}
```

A) 0 1 2
   0 1 2
   0 1 2

B) 0 0 0
   1 1 1
   2 2 2

C) 1 1 1
   1 1 1
   1 1 1

D) 1 0 0
   0 1 0
   0 0 1

15. Which one of the following statements are true (if any) ?
I A class can only have one constructor
II A class cannot have two methods with the same name

A) I only
B) II only
C) Both I and II
D) Neither I nor II
16. Given that B is a subclass of A and C is a subclass of B, what is printed by the following code segment?

```java
A a = new B();
if (a instanceof A)
    System.out.println("yes1 ");
if (a instanceof B)
    System.out.println("yes2 ");
if (a instanceof C)
    System.out.println("yes3");
```

A) yes1  
B) yes2  
C) yes1 yes2  
D) yes1 yes2 yes3

17. In order to promote encapsulation (data-hiding), which visibility modifier should be used for instance variables?

A) public  
B) protected  
C) default (no modifier)  
D) private

18. In the program below, which one of the following println() statements will result in a compilation error?

```java
public class TestScope
{
    public static void main (String [] args)
    {
        int a = 10;
        System.out.println("1: a = "+a);

        for (int b = 0; b < 10; b++)
        {
            System.out.println("2: a = "+a+" b "+b);
            int c = 1;
            a = a + c;
            System.out.println("3: a = "+a+" b "+b+
"c = "+c);
        }
        System.out.println("4: a = "+a+" b "+b+
"c = "+c);
    }
}
```
A) First println() statement (labelled 1:)
B) Second println() statement (labelled 2:)
C) Third println() statement (labelled 3:)
D) Fourth println() statement (labelled 4:)

19. Which one of the following statements is false?
A) An abstract class must have an abstract method
B) An abstract class can have instance variables
C) An abstract class can have a constructor
D) An abstract class cannot be instantiated

20. Which of the following statements is false in regards to files?
A) You can move directly to any point in a binary file using the seek() method
B) If a program does not close a text file after it has finished writing out data, then the data will not be written out correctly
C) An exception is thrown if a text file that does not exist is opened for reading
D) An exception is thrown if a text file that does not exist is opened for writing
### C.2 Individual Responses to Multiple Choice Questions

*Table C.1: Individual Responses to Multiple Choice Questions (Table 1)*

<table>
<thead>
<tr>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
<th>q5</th>
<th>q6</th>
<th>q8</th>
<th>q9</th>
<th>q11</th>
<th>q12</th>
<th>q14</th>
<th>q15</th>
<th>q16</th>
<th>q17</th>
<th>q18</th>
<th>q19</th>
<th>Total</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s15</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s16</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s18</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s19</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s20</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s21</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s22</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s23</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s24</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s25</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s26</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s27</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s28</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s29</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s30</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s31</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s32</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s33</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s34</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>s35</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
## Table C.2: Individual Responses to Multiple Choice Questions (Table 2)

<table>
<thead>
<tr>
<th></th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
<th>q5</th>
<th>q6</th>
<th>q8</th>
<th>q9</th>
<th>q11</th>
<th>q12</th>
<th>q14</th>
<th>q15</th>
<th>q16</th>
<th>q17</th>
<th>q18</th>
<th>q19</th>
<th>Total</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>s36</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>0.63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s37</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>s38</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>s39</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>s40</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>s41</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>s42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>s43</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0.31</td>
</tr>
<tr>
<td>s44</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>s45</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>s46</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>s47</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0.25</td>
</tr>
<tr>
<td>s48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.19</td>
</tr>
<tr>
<td>s49</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>0.88</td>
</tr>
<tr>
<td>s50</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0.63</td>
<td></td>
</tr>
<tr>
<td>s51</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>s52</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>s53</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>s54</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>0.50</td>
<td></td>
</tr>
<tr>
<td>s55</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>s56</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>s57</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0.38</td>
<td></td>
</tr>
<tr>
<td>s58</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>s59</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>s60</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>s61</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0.44</td>
</tr>
<tr>
<td>s62</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>s63</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td>0.63</td>
</tr>
<tr>
<td>s64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0.13</td>
</tr>
<tr>
<td>s65</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0.44</td>
</tr>
<tr>
<td>s66</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>s67</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>s68</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>10</td>
<td>0.63</td>
</tr>
<tr>
<td>s69</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>0.44</td>
</tr>
<tr>
<td>s70</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>0.63</td>
<td></td>
</tr>
</tbody>
</table>
### Table C.3: Individual Responses to Multiple Choice Questions (Table 3)

<table>
<thead>
<tr>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
<th>q5</th>
<th>q6</th>
<th>q8</th>
<th>q9</th>
<th>q11</th>
<th>q12</th>
<th>q14</th>
<th>q15</th>
<th>q16</th>
<th>q17</th>
<th>q18</th>
<th>q19</th>
<th>Total</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>s71</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>53</td>
<td>0.60</td>
</tr>
<tr>
<td>s72</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>55</td>
<td>0.62</td>
</tr>
<tr>
<td>s73</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>55</td>
</tr>
<tr>
<td>s74</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>58</td>
</tr>
<tr>
<td>s75</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>46</td>
<td>0.48</td>
</tr>
<tr>
<td>s76</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>43</td>
<td>0.48</td>
</tr>
<tr>
<td>s77</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>41</td>
</tr>
<tr>
<td>s78</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>31</td>
</tr>
<tr>
<td>s79</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>31</td>
</tr>
<tr>
<td>s80</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>28</td>
</tr>
<tr>
<td>s81</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>24</td>
</tr>
<tr>
<td>s82</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>24</td>
</tr>
<tr>
<td>s83</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>13</td>
<td>21</td>
</tr>
<tr>
<td>s84</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>s85</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>s86</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>s87</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>s88</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>s89</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>53</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>48</td>
<td>57</td>
<td>58</td>
<td>46</td>
<td>43</td>
<td>41</td>
<td>31</td>
<td>31</td>
<td>33</td>
<td>84</td>
<td>60</td>
<td>39</td>
<td>Average 0.60</td>
</tr>
<tr>
<td></td>
<td>0.62</td>
<td>0.64</td>
<td>0.52</td>
<td>0.46</td>
<td>0.35</td>
<td>0.37</td>
<td>0.67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### C.3 Examination Questions for Semester 1, 2008

#### C.3.1 Part 1

**Instruction:** For the multiple-choice questions, select the most appropriate response and circle the corresponding letter (A/B/C/D). There is only one correct answer for each question.

1. Which of the following is not a primitive type in Java?

   A) boolean  
   B) byte    
   C) String  
   D) float
2. What will be the value assigned to the variable \( x \) as a result of the following statement?
\[
\text{int } x = 2 \div 10 \div 2 \div 4 \div 5;
\]
A) 5
B) 7
C) 8
D) 10

3. Which one of the given sets of \( X \), \( Y \), and \( Z \) values makes the following expression true?
\[
\text{! ( } X \text{ && } Y \text{ ) && ( } \neg Y \text{ || } Z \text{ )}
\]
A) \( X = \text{false}, Y = \text{false}, Z = \text{false} \)
B) \( X = \text{false}, Y = \text{true}, Z = \text{false} \)
C) \( X = \text{true}, Y = \text{true}, Z = \text{false} \)
D) \( X = \text{true}, Y = \text{true}, Z = \text{true} \)

4. Given that \( x \) and \( y \) are both variables of type int, the statements:
\[
y = x + x + x;
y += y + x;
\]
are equivalent to:
A) \( y = 4 \times x \)
B) \( y = 5 \times x \)
C) \( y = 7 \times x \)
D) \( y = 8 \times x \)

5. What will be assigned to the variable \( \text{result} \) after execution of the statements below, if the value that is entered by the user is -1?
\[
\text{int } \text{something} = \text{console.nextInt();}
\text{String result;}
\text{if (something} <= 0)
\text{\hspace{1em}}\text{result = "ONE";}
\text{else if (something} <= 50)
\text{\hspace{1em}}\text{result = "TWO";}
\text{if (something} <= 100)
\text{\hspace{1em}}\text{result = "THREE"}
\text{else}
\text{\hspace{1em}}\text{result = "FOUR";}
\]
A) "ONE"
B) "TWO"
C) "THREE"
D) "FOUR"
6. What is the output of the following code segment?

```java
for (int i = 1; i <= 15; i += 2) {
    if (i % 3 == 0)
        System.out.print("a");
    else
        System.out.print("b");
}
```

Answer:

7. What is the output of the code segment below?

```java
String message = "Sally is counting: ";
String numbers = "";
for (int i = 0; i < 12; i += 2)
    numbers = numbers + i + " ";
System.out.println(message + numbers);
```

A) Sally is counting: 0 2 4 6 8 10
B) Sally is counting: 0 1 2 3 4 5 6 7 8 9 10 11
C) Sally is counting: 0 2 4 6 8 10 12
D) Sally is counting: 0 1 2 3 4 5 6 7 8 9 10 11 12

8. What will be the output of the program segment below if marks is input as 50?

```java
System.out.print("Enter marks : ");
int marks = console.nextInt();
if (marks < 50)
    System.out.print(" Fail");
if (marks >= 50)
    System.out.print(" Pass");
else if (marks >= 60)
    System.out.print(" Credit");
if (marks <= 70)
    System.out.print(" Distinction");
else
    System.out.print(" High-Distinction");
```

A) Pass Credit
B) Pass Distinction
C) Pass High-Distinction
D) Pass Credit Distinction High-Distinction
9. How many times will the do-while loop below be executed?

```java
int m = 0;
    do {
        System.out.println(m);
        m = m - 1;
    } while (m > 0);
```

A) 0 times
B) 1 time
C) 10 times
D) It is an infinite loop (ie. it will never stop executing)

10. Which of the following statements is false?

A) A static method can be accessed directly via the class without having to create an object from that class first (visibility permitting)
B) A non-static (instance) method cannot refer to a static variable inside the same class
C) A static method cannot refer to non-static (instance) variable inside the same class
D) All objects created from a class which includes a static variable may change the value stored in that static variable (visibility permitting)

11. What are the values of the variables a and b after the execution of the following program:

```java
public class TestParameterPassing {
    public static void main (String [] args) {
        double a = 2.5;
        String b = "Hello";
        anyMethod(a, b);
    }
    public static void anyMethod (double d, String s) {
        d = d * 3;
        s = "Goodbye";
    }
}
```

A) a = 2.5, b = "Goodbye"
B) a = 2.5, b = "Hello"
C) a = 7.5, b = "Goodbye"
D) a = 7.5, b = "Hello"
12. Which one of the following statements is true?

A) An abstract class must have an abstract method  
B) All methods in an abstract class must themselves be abstract  
C) An abstract class cannot define instance variables  
D) An abstract class cannot be instantiated

13. Which of the following statements is true in regards to files?

A) You cannot write numeric values to a text file  
B) A text file that has been opened for writing must be closed after the program has finished writing data in order for the data to be written out correctly.  
C) No exception is thrown if a text file that does not exist is opened for reading  
D) An exception is thrown if a text file that does not exist is opened for writing

14. Given that B is a subclass of A and C is a subclass of B, what is printed by the following code segment?

```java
A a = new C();
if (a instanceof A)
    System.out.print("yes1 ");

if (a instanceof B)
    System.out.print("yes2 ");

if (a instanceof C)
    System.out.print("yes3");
```

A) yes1  
B) yes2  
C) yes1 yes2  
D) yes1 yes2 yes3

15. What is the output of the program below?

```java
public static void main (String[] args) {
    int[][] m = new int[3][3];

    for (int row = 0; row <= 2; row++)
        for (int col = 0; col <= 2; col++)
            m[row][col] = col;

    for (int i=0; i<3; i++)
        {
```
for (int j=0; j<3; j++)
    System.out.print(" "+m[i][j]);
System.out.println();
}
}
A) 0 1 2
   0 1 2
   0 1 2
B) 0 0 0
   1 1 1
   2 2 2
C) 2 1 0
   2 1 0
   2 1 0
D) 2 2 2
   2 2 2
   2 2 2

16. What is the output produced by the following code segment (assuming that the exception
types ExceptionTypeOne and ExceptionTypeTwo have been defined previously and are not
related by inheritance)?

public void aMethod()
{
    int value = 10;
    try
    {
        System.out.print("one ");
        if (value <= 10)
        {
            System.out.print("two ");
            throw new ExceptionTypeTwo();
            System.out.print("three ");
        }
        catch (ExceptionTypeOne e)
        {
            System.out.print("four ");
        }
        finally

APPENDIX C. NOVICE PERSPECTIVES SURVEY

```java
{ 
    System.out.print("five ");
} 
System.out.print("six ");
}
```

A) one two
B) one two five
C) one two three five six
D) one two four five six

17. If the value of the variable x is -10, 10 and 100 respectively then what is printed after the execution of the following code segment for each of the values?

```java
if (x > 50 && x > 0) 
    System.out.println("Success!");
else 
    System.out.println("Failure!");
```

A) Failure!, Failure!, Success!
B) Failure!, Success!, Failure!
C) Failure!, Failure!, Failure!
D) Success!, Success!, Failure!

18. Consider the following code segment?

```java
int[] x = {2, 1, 4, 5, 7};
int limit = 3;
int i = 0;
int sum = 0;

while ((sum < limit) && (i < x.length))
{
    ++i;
    sum += x[i];
}
```

What value is in the variable "i" after this code is executed?
A) 0
B) 1
C) 2
D) 3
19. Consider the following code segment:

```java
int[] x = {0, 1, 2, 3};
int temp;
int i = 0;
int j = x.length-1;

while (i < j)
{
    temp = x[i];
    x[i] = x[j];
    x[j] = 2*temp;
    i++;
    j--;
}
```

After this code is executed, array "x" contains the values:
A) 3, 2, 2, 0
B) 0, 2, 2, 3
C) 0, 2, 4, 6
D) 6, 4, 2, 0

20. Consider the following code segment:

```java
int[] x1 = {1, 2, 4, 7};
int[] x2 = {1, 2, 5, 7};
int i1 = x1.length-1;
int i2 = x2.length-1;
int count = 0;

while ((i1 > 0) && (i2 > 0))
{
    if (x1[i1] == x2[i2])
    {
        ++count;
        --i1;
        --i2;
    }
    else if (x1[i1] < x2[i2])
    {
        --i2;
    }
    else
    {  // x1[i1] > x2[i2]
        --i1;
    }
```
C.3.2 Part 2

Instruction: Write your answers in the space provided

Question 24
Complete the HighLow class below to identify and display the highest and lowest of the series of positive integer values passed into the program as command line arguments.

Expected Input/Output is shown below.
java HighLow 7 4 9 10
Highest value passed in was 10
Lowest value passed in was 4

java HighLow 45 52 81 69 23 97 76
Highest value passed in was 97
Lowest value passed in was 23

Notes:
Command-line arguments are passed to the main method through the array of String references (args in the main method below). The size of any array can be accessed through its length attribute (note: you can assume at least one valid argument will be passed in on the command line). You will need to use Integer.parseInt() to convert each command line argument to integer format before processing it.

(5 marks)

```java
public class HighLow {
    public static void main (String[] args)
    {
        int highestArg = 0;
        int lowestArg = 0;
        int nextArg;
        // Write your code here
    }
    // Write your code here
}
```

A) 3
B) 2
C) 1
D) 0
System.out.println( "Highest value passed in was " + highest);
System.out.println( "Lowest value passed in was " + lowest);
}
Appendix D

Instructor Views Questionnaire

This appendix presents the questionnaire distributed to the instructors and the raw data for their responses. The link to the survey is:
D.1 Instructor Questionnaire

In this section we include the Project Information Statement and the survey questions for the programming instructors.
Invitation to Participate in a Research Project

Project Information Statement

Project Title:
- Debugging: A constructivist approach to learning programming

Investigators:
- Miss Shuhaida Mohamed Shuhidan, (PhD student, School of CS&IT, RMIT University)
  shuhaida.mohamedshuhidan@rmit.edu.au, (03) 99252758
- Dr Margaret Hamilton, (Senior Lecturer, School of CS&IT, RMIT University)
  margaret.hamilton@rmit.edu.au, (03) 99252939
- Dr Daryl D’Souza, (Senior Lecturer, School of CS&IT, RMIT University)
  daryl.dsouza@rmit.edu.au, (03) 99252947

Dear programming instructors,

You are invited to participate in a research project being conducted by RMIT University. This information sheet describes the project in straightforward language, or ‘plain English’. Please read this sheet carefully and be confident that you understand its contents before deciding whether to participate. If you have any questions about the project, please ask one of the investigators.

In the remainder of this document, the term instructors denotes the university academic members who are involved in teaching Introduction to Programming (IP) or Programming 1 (P1) or Programming 2 (P2) or Programming 3 (P3).

Who is involved in this research project? Why is it being conducted?

Instructors (including RMIT staffs and other institutions) and investigators will be involved in this project.

The research seeks to improve learning programming by novices through an investigation of instructors’ perspectives of programming questions in final examinations.

Why have you been approached?

We are conducting this research to enhance novices ability to learn programming. We are approaching you as instructors who are, or have been in the past, responsible for teaching programming in Introduction to Programming (IP) or P1, P2 or P3. All data will remain confidential.

What is the project about? What are the questions being addressed?

High attrition rates are often experienced in Computer Science schools, in part due to students’ inability to learn programming, a core survival skill. Novices find it difficult to grasp the foundation level concepts easily, resulting in grief and frustration, and ultimately, surrender. Yet, those who manage to overcome their learning difficulties are able to move on, even excel.

We would like to consider the instructor’s expectations of novices, as evidenced by their exam-paper questions. In particular, we are interested in finding out why instructors use multiple-choice questions, and what information they would hope such questions would provide them. If an instructor would like to learn about the breadth of novice ability, or the depth of their knowledge, would that be tested by multiple choice questions, or short answer questions, or coding questions, or combinations of all three?

If I agree to participate, what will I be required to do?

You will be asked to answer few questions on how you devise questions (Section A) and to categorise the level of complexity of the questions based on the guide given (Section B). There is also room to explain your responses or provide any useful insights into teaching novice programmers.

What are the risks or disadvantages associated with participation?

Any participation will remain voluntary and anonymous. Thus, it can be stated that there is no risk to you for being involved in this particular study.

What are the benefits associated with participation?

Your participation will help us to further to improve the learning of programming. We aim to develop a tool to help novices in learning programming based on a fundamental understanding of where the novices are experiencing difficulty.

What will happen to the information I provide?

Any information you provide will be totally anonymous and will remain confidential. Only the members of the investigative team will see such data, which will be kept secure for a period of five years and then destroyed.

Because of the nature of data collection, we are not asking you to provide written informed consent of participation in the project.

Instead, we assume that you have given consent by clicking the button ‘Yes, I agree to participate’.

227
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

What are my rights as a participant?

Your rights as a participant of this project include:

- The right to withdraw their participation at any time, without prejudice.
- The right to have any unprocessed data withdrawn and destroyed, provided it can be reliably identified, and provided that so doing does not increase any risk for you.
- The right to have any questions answered at any time.

Whom should I contact if I have any questions?

Any member of the investigative team listed at the beginning of this plain language statement may be contacted at any time. Any complaints about the conduct of this research project can be made to the Executive Officer, RMIT Human Research Ethics Committee, see http://www.rmit.edu.au/rd/hrec_complaints and the footer of this page.

Yours Sincerely,

Ms Shuhaida Mohamed Shuhidan, MSc
Dr Margaret Hamilton, PhD
Dr Daryl DSouza, PhD

Yes, I agree to participate  No, I disagree to participate

Any complaints about your participation in this project may be directed to the Executive Officer, RMIT Human Research Ethics Committee, Research & Innovation, RMIT, GPO Box 2476V, Melbourne, 3001.

Details of the complaints procedure are available from the above address or http://www.rmit.edu.au/rd/hrec_complaints
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

Survey for Instructor

Instructions: Please answer all 11 questions in the survey. The purpose of this survey is to understand the instructors’ perspectives regarding devising questions in summative assessment (a final assessment of a module). There are no right or wrong answers. The researchers are interested in your views and opinion; we hope that the survey will help to improve teaching and learning of programming courses.

Section A: Instructor Background and Perspectives

In this section, please fill in the blanks and select the responds, where appropriate.

1. How many semesters have you been:
   lecturing/teaching/tutoring programming? __________________ semester(s)
   devising questions for the programming exam? __________________ semester(s)

2. Which level of Programming courses have you covered?
   (You can select more than one respond that describe you best)
   ☐ IP : This course covers foundation of computer system, conceptual building block for programming, usually for non-Computer Science or IT related major (Zero prerequisite)
   ☐ P1 : This course covers introductory programming concepts for novices (IP or Zero prerequisite)
   ☐ P2 : This course extends the learning of programming to cover more difficult programming principles (P1 prerequisite)
   ☐ P3 : This course covers a detailed study of programming; including the use of defensive programming, debugging, testing, coding standards and practices. (P1 and P2 prerequisite)

3. Why do you use multiple choice questions as an instrument for summative assessment?

   Strongly Disagree Disagree Neutral Agree Strongly Agree
   Easy to mark
   Precise answer provided
   To give some hints of the fragment’s code to the next section of the assessment
   To test students on a low level of understanding question
   Other (please specify): ____________________________

4. How confident are you that multiple choice questions in the final exam will do any of the following:

   No confidence at all Not confident Neutral Confident Completely confident
   Test the student’s understanding of programming concepts
   Encourage the students to think carefully to select the best answer
   Encourage the students to guess the answer
   Other (please specify): ____________________________

5. How do you devise the distractors?

   Definition for distractors: the options other than the answer for multiple choice questions.

6. What factors do you consider when you devise exam questions?

   Strongly Disagree Disagree Neutral Agree Strongly Agree
   The question needs to cover a particular programming topic

229
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

The question needs to have a certain level of difficulty  

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other (please specify):

7. Do you think summative assessment is a valid measure of a student's ability to program?

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Section B: Instructor Levels of Complexity

In this section, please indicate the level the complexity you consider the questions based on basic measures given (low, medium, high). Please respond on the base that the questions are intended to be tested for students in introductory programming course.

Definition for **Syntax Knowledge**: Knowledge regarding the language; This includes the grammar and flow of the language.

Definition for **Semantic Knowledge**: Knowledge relating to the interpretation of the language constructs.; This includes concepts, principles, rules and problem solving strategies.

Definition for **Problem solving skills**: A measure of the problem solving skills needed to solve the question for an average student.

Definition for **Level of difficulty**: A measure for complexity of the question and the number of skills or concepts tested.

**Question 8**

```java
public class Reftest {
    public static void main (String args[ ] )
    {  
        String s;
        s = new String("Apple");
        s = new String("Orange");
        s = new String("Banana");
        System.out.println ("s is now referring to = "+s);
    }
}
```

What is the output of the program?

A. s is now referring to Apple  
B. s is now referring to Orange  
C. s is now referring to Banana  
D. s is now referring to Apple Orange Banana

<table>
<thead>
<tr>
<th>Skills/knowledge tested</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax knowledge</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic knowledge</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem solving skills</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difficulty of problem</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Question 9**

Consider the following code fragment:

```java
int[ ] x1 = {0, 1, 2, 3};
int[ ] x2 = {1, 2, 2, 3};
int i1 = 0;
int i2 = 0;
int count = 0;
while ( (i1 < x1.length) && (i2 < x2.length) )
{  
    if ( x1[i1] == x2[i2] )
    {  
        ++count;
        ++i2;
    }
    else if (x1[i1] < x2[i2])
    {  
        ++i1;
    }
    else
    {  
        // x1[i1] > x2[i2]
        ++i2;
    }
}
```

Skills/knowledge tested  

<table>
<thead>
<tr>
<th>Syntax knowledge</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic knowledge</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem solving skills</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difficulty of problem</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

230
After this code is executed, "count" contains:

a) 0
b) 1
c) 2
d) 3
e) 4

Skills/knowledge tested

<table>
<thead>
<tr>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax knowledge</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Semantic knowledge</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Problem solving skills</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Difficulty of problem</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Question 10

The lines of code provided below are jumbled. When the lines are correctly ordered, the code calculates the average of the numbers stored in the array "a".

```java
average = 0;
count++; sum = 0;
average = sum / Count;
else {
while (Count < x.length) {
count = 0;
sum = sum + x[Count];
if (Count > 0) {
}
}
}
```

When the code is correctly ordered, in what order would some of the lines occur?

a) sum = 0 before if (Count > 0) { before sum = sum + x[Count];
b) sum = 0 before count++ before if (Count > 0) {
c) sum = sum + x[Count]; before sum = 0 before count++
d) sum = sum + x[Count]; before count++ before sum = 0

Skills/knowledge tested

<table>
<thead>
<tr>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax knowledge</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Semantic knowledge</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Problem solving skills</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Difficulty of problem</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Question 11

Suggest a name for method10 below that reflects its purpose:

```java
public float method10(int[] aiNumbers)
{
int iSum = 0;
for (int iLoop = 0; iLoop < aiNumbers.length; iLoop++)
{
iSum += aiNumbers[iLoop];
}
return iSum / aiNumbers.length;
}
```

Skills/knowledge tested

<table>
<thead>
<tr>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax knowledge</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Semantic knowledge</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Problem solving skills</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Difficulty of problem</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
D.2 Instructor Responses

This section shows the raw data collected from 66 programming instructors. In the first line, the numbers represent the data collected from the multiple choices responses. For example, the record for *instructor1* is 1818111533322444212122212221222. These figures represent instructors’ responses of the multiple choice questions to Question 1, Question 2, Question 3, Question 4, Question 6, Question 7, Question 8, Question 9, Question 10 and Question 11. It means 18 semesters of lecturing or teaching or tutoring programming courses, 18 semesters of experience in devising questions for programming exams, 1 for yes the respondent taught/is teaching IP, P1, P2 and P3 (0 for has not taught or teaching) followed by any of 5 for strongly agree, 4 for agree, 3 for neutral, 2 for disagree, 1 for strongly disagree for the remaining questions in Section A of the questionnaire, and 3 for high, 2 for medium and 1 for low for the responses in Section B of the questionnaire.

Q3 is the instructors open ended responses that were included in other (please specify)’s column for Question 3 of the questionnaire. Other (please specify)’s column is optional for the instructors to fill in. Q4 represents Question 4, and so do Q5 and Q6 that represent Question 5 and Question 6.

*instructor1
1818111533322444212122212221222
Q3: 
Q4: 
Q5: to make a certain error and follow it.
Q6: 

*instructor2
202001140342354522211232222222211
Q3: 
Q4: 
Q5: Permuting the correct values (input / output) to reach ”plausible” but incorrect values
Q6: 

*instructor3
410101542544254112112222222222
Q3: 
Q4: 
Q5: I use things that are “obviously” wrong if you have any clue as to the real answer. There are no tricks on my questions. I simply take answers from my other test questions that make no sense for this test question.
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

Q6:

*instructor4
30301100431444232331123232322322
Q3:
Q4:
Q5: based on common misconceptions
Q6:

*instructor5
4848011553145344222232323321222
Q3:
Q4:
Q5: In the case of numerical questions (e.g. “the number of bytes required to store an RGB encoded pixel”), use numbers that are answers to other related questions (e.g. 8, 256, 17million). In cases of terminology (e.g. “a class that has one or more methods without method bodies”), use items that are plausible (e.g. “interface”, “public class”, “static class”). In cases where calculation or reasoning is required, I use insights from previous student discussions and frequent errors in non-multiple choice questions.
Q6:

*instructor6
440100534013543111122222321111
Q3:
Q4:
Q5: Common incorrect answers seen in tutorials
Q6:

*instructor7
14170115414545541312122313221111
Q3:
Q4:
Q5: I try to devise a distractor for each conceptual “bug” students tend to exhibit.
Q6:

*instructor8
525211112204454554221121131131122
Q3: To test situations where students would normally have other cues not provided on the test (like IDE contents)
Q4:
Q5: Look for the answers which would be provided by common mistakes
Q6:

*instructor9
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

108011000000005441111221232122212212
Q3: Never use
Q4: Never use
Q5: Never use multiple choice questions
Q6:

*instructor10
101001000000000331232322222222222
Q3: I never use multiple choice questions.
Q4: I feel that multiple choice is a completely inappropriate tool for judging deep understanding and comprehension of programming concepts
Q5: I don’t
Q6:

*instructor11
7454111154234444322112213221321221
Q3: I rarely give multiple choice questions about programming
Q4: Good multiple choice questions can do these things.
Q5: I think about students’ misconceptions and I apply them to the problem to get accordingly mistaken answers. And sometimes I make up something funny for the fifth one. Research shows that the best number of possible responses is actually three, not five.
Q6:

*instructor12
2424111054342345431211221222222222
Q3:
Q4:
Q5: I very seldom use multiple choice questions on programming exams
Q6:

*instructor13
181801115444423543322133323332322
Q3:
Q4:
Q5: I use familiar-sounding terminology, terminology from other areas of the course, and sometimes humor
Q6:

*instructor14
3030111000003244431221232223332332
Q3: I don’t.
Q4:
Q5: I don’t use multiple choice.
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

Q6:

*instructor15
302811115555443534111123212321211
Q3: To constrain the students creativity
Q4:
Q5: I would rather not think of them as distractors. Typical errors, attractive mistakes, close but not there
Q6: It can be solved in a limited time frame, without recourse to a computer

*instructor16
504511115534442544121122222222211
Q3:
Q4:
Q5: I think of other things we have talked about that might be somewhat related (for example, selection in a question on iteration). I think of answers that are so outrageously incorrect that no student would ever select them - but still some do (for example, computers use binary because they have only two fingers).
Q6:

*instructor17
1616111134354434441111222212221111
Q3:
Q4:
Q5: Based on the questions the students have asked in class and the various ways the students could make mistakes.
Q6:

*instructor18
1010011143143343301221222211121222
Q3:
Q4:
Q5: Answers that are almost correct, as well as one's that are way off the mark.
Q6:

*instructor19
1010011054114534441111223323222222
Q3: encourage students to analyze different responses noting differences and choosing the best alternative
Q4:
Q5: by choosing answers that are likely to be picked by students with common misunderstandings
Q6:
Q3: It depends on the question. My distracters tend to fall into two categories: Those really close to the correct answer, yet slightly different, and those unlike the correct answer, yet sounding reasonable. The more precise the answer has to be, the more likely the distracter will be in the former category.
Q6: You need to include essay questions because CS students need to know how to write.

Q3: I don’t use multiple choice
Q4: I don’t use multiple choice
Q5: Use terms or concepts that are similar, or use other terms that have also been used frequently. Students who are playing “buzzword bingo” without understanding what the terms mean will be more likely to be distracted by them. Ideally, distracters should also include one or two items that are ‘close’ to the correct answer, i.e. require some thought or careful attention to terms to distinguish from correct answers.

Q3: Do not use multiple-choice questions.
Q4: N/A
Q5: I would use known misconceptions to form some of them and typically some “almost correct” ones.
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

2626011134443424431111333322222222
Q3: I try to come up with plausible sounding distracter that “sound like” or “look like” or are “easily confused with” the real answers. This catches those who are guessing, don’t know but think they do, or who are careless.
Q6: coverage ... I try to include enough questions to cover all the important concepts

*instructor26
12120110322222224442121122122222222
Q3: Give common misperceptions as possible answers to help define exact level of knowledge
Q4: After defining the purpose of the question, define all possible common errors that students may make when examining the code; these become the distracters.
Q6:

*instructor27
18181111000000053223123323333333
Q3: I do not use multiple choice questions on my programming course exams.
Q4: Although I use them, I regard mc quests as “free points” – not really diagnostic, but keeps the weaker students on track.
Q5: Ideally, the question should remind students of something they know and lead them to the correct answer. With many questions, the answer is just a list of numbers, with only one correct. The others are not really “distracters”, just wrong. Questions that have tempting-but-wrong answers are sometimes called “trick questions” and should not be used very often. But of course there are some ”trick situations” in programming, like integer division or operator precedence, which have to be tested with likely wrong answers.
Q6:

*instructor28
2020011152414445441111232223221222
Q3: Students have been trained in high school (in the US, at least) to expect m-choice q’s and are happy to see them. Keeping students happy is important.
Q4: Although I use them, I regard mc quests as “free points” – not really diagnostic, but keeps the weaker students on track.
Q5: Ideally, the question should remind students of something they know and lead them to the correct answer. With many questions, the answer is just a list of numbers, with only one correct. The others are not really “distracters”, just wrong. Questions that have tempting-but-wrong answers are sometimes called “trick questions” and should not be used very often. But of course there are some ”trick situations” in programming, like integer division or operator precedence, which have to be tested with likely wrong answers.
Q6:

*instructor29
4040111145442244421211223233331221
Q3: Give common misperceptions as possible answers to help define exact level of knowledge
Q4: After defining the purpose of the question, define all possible common errors that students may make when examining the code; these become the distracters.
Q6:
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

*instructor30
363611111131114114222122222122311
Q3: I rarely use multiple choice as a method for testing; when I do it is a quiz to ensure students are reading
Q4:
Q5: from experience, I generally know how students will often misunderstand a problem, or (mis)code a solution – I sometimes get this from in class formative assessments like pop-group-assignments to solve a problem, or interactive learning (think-pair-share) Q6: should complement other questions, even test again to ensure the student did not just get lucky on one question – I also ask questions directly from the text/lecture

*instructor31
about 6565111100000005551111121222221211
Q3: I do not use multiple choice questions in my programming courses.
Q4: I do not use multiple choice questions in my programming courses.
Q5: I do not devise distractors – free response questions seem more appropriate to me.
Q6: I think free response questions have high validity; I have much less confidence in multiple choice questions.

*instructor32
24241110433343432312332333333222
Q3: I don’t use multiple choice questions.
Q4:
Q5: I would base them on observed errors/typical mistakes that occur when students have a concept confused.
Q6:

*instructor33
880100553544444312222323233233232
Q3:
Q4: Some think carefully; some guess.
Q5: Usually, variations on the right answer. If the question is “what will this loop print out”, a distractor might stop one integer value too early (or late), etc
Q6: Have I used this question before? If so, need to alter it for a second use. (What, BTW, is “summative assessment”?)

*instructor34
84111054113245521211232223331221
Q3: I do not use multiple-choice questions.
Q4:
Q5:
Q6:

*instructor35
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

Q3: I rarely use multiple choice so answers above should be filtered through that lens.
Q4:
Q5: Use common errors that students make. When there are limited number of answers, e.g., big-oh, list all possible reasonable answers.
Q6:

*instructor36
403811000000224544121122232132312
Q3: I never use multiple choice questions.
Q4:
Q5:
Q6:

*instructor37
2020011023244335421321233223332222
Q3: To test understanding of fundamental terms/phrases used in programming.
Q4:
Q5: Use words that sound similar to the correct answer, use terms/phrases that are discussed in class but are not related to the question, use words/phrases that have nothing to do with computer science.
Q6:

*instructor38
1100103333234554111112112112112211
Q3:
Q4:
Q5: I don’t use multiple choice questions
Q6:

*instructor39
3230011100002345441211233200000000
Q3: I have NEVER used multiple choice questions in an exam!
Q4:
Q5: I have NEVER used multiple choice questions in an exam!
Q6: Not sure how to interpret these questions!! What question does not cover a particular programming topic?

*instructor40
262611105545353553111222322331111
Q3:
Q4:
Q5: Obvious way: introduce single errors that make this the wrong answer.
Q6:
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

*instructor41
1010011055442235521111221322222222
Q3: Um, I don’t use multiple choice.
Q4:
Q5: I don’t use multiple choice.
Q6:

*instructor42
2018011100003341141211122022321211
Q3: I do not use multiple choice questions on exams for programming courses
Q4:
Q5: I do not use multiple choice questions
Q6: I try to ensure the student learned certain concepts from class and from outside projects

*instructor43
1717111155555555551312123312331322
Q3: multiple choice can also be used to test the depth of knowledge
Q4: It depends on how you weight wrong answers
Q5: using common misconceptions
Q6:

*instructor44
202011005555244553332233333332222
Q3:
Q4:
Q5: I try to think of possible answers that are incorrect.
Q6:

*instructor45
10101111551144352211113322333332222
Q3:
Q4:
Q5: Try to come up with something that is wrong but not too obviously wrong. I try not to be tricky or sneaky but expect the student to know the answer.
Q6: The question shouldn’t be hard - just the material it covers.

*instructor46
105111044353034431111121222221211
Q3:
Q4:
Q5: - devise propositions that might seem right, but are not for at least on important reason
- devise results based on common misconceptions
Q6:
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

*instructor47
18-2018-20110543433512112222322111
Q3: Shorter feedback time
Q4:
Q5: Depending on the complexity of the question, the distracters are either very close or opposite. I try to encourage careful thought.
Q6: How much foundational knowledge is necessary and how much application rather than cognition.

*instructor48
44011055414245542311333323322311
Q3:
Q4:
Q5: Use the same “wrong” syntax at least 2 times. Provide answers such as “none of the above”, “all of the above”, “only A and C”, “only A and D”
Q6:

*instructor49
3000100555441545222000000000000
Q3:
Q4:
Q5: Utter nonsense for at least one. A distracter could be the dog, cat, or mouse.
Q6:

*instructor50
4201105543455432221232233322322
Q3:
Q4:
Q5: I try to come up with choices that reflect the common mistakes students would make, such as confusing terms or concepts (e.g. polymorphism vs. encapsulation vs. inheritance), forgetting that integer arithmetic truncates, off-by-one errors in loops, etc. You want to make sure that the correct isn’t the only answer that would work, and that the student has to apply concepts they’ve learned in order to rule out the incorrect answers.
Q6:

*instructor51
440100351145345411112212222222121
Q3:
Q4:
Q5: I use common misconceptions of the topic or other common errors made during the process or procedural concept being tested. I do not advocate the use of distracters that detect whether or not the tester has read the question carefully during the examination, as I am not trying to evaluate the student’s ability to test well. Examples of distracters I do not
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

use are those that are very similar to the correct option in phraseology.
Q6:

*instructor52
100111124224424521321123213331222
Q3: I think it provides many options for students and they can decipher which is the right one - gets them thinking. Also, the format of the AP CS exam is multiple choice; so this prepares them for that.
Q4: I take the right answer and then create similar ones close to that but I also put other distracters that would have something to do with another issue in programming - just so they know the difference. A lot of times, the multiple choice questions I use are already created by other people - previous AP Exams, AP Study guides, etc.
Q6:

*instructor53
202011115344243543111122321221111
Q3: try to think of answers that might seem logical but would be wrong. Or, think through incorrect solutions to the problem and see what that incorrect solution would generate as an answer.
Q6:

*instructor54
1414111144354325543211322231322211
Q3: part of the use of multiple choice questions is to prepare students for later courses where these will be even more common
Q5: If the question is based on mathematics, by making common math errors. Otherwise, by previous student mistakes. If the students don’t understand, the mistakes are often repeated.
Q6: The material has been covered adequately in the course.

*instructor55
242411105415325543221122123323231
Q3: I don’t any multiple choice questions on an exam. I do use them on weekly quizzes on the first few quizzes. I usually use multiple choice questions for definition of terms problems, in which case I use other terms covered during the same time period. I also use multiple choice questions for “step-through” problems. The distracters in that case would represent the various wrong ways that I’ve seen students step through the code.
Q6:
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

Q3: From experience marking student assignments: I use their common mistakes.
Q4:
Q5: give weaker students confidence; test minimal/basic knowledge; prep them during semester using MCQ
Q6: test specific theory; lead up to final assessment as I use these during course too.
Q5: I have at most 1 or two funny distracters as these really reduce the multiple choice; statistically they hardly chosen and would only reveal the clueless. I use possibilities that could be logical and force students to think, calculate or at least prepare extremely well (lots of factual knowledge has some value too). But the important problem solving I hardly mix up with multi-choice.
Q6: The student MUST be able to determine how much TIME to commit and the TIME must be really short.
APPENDIX D. INSTRUCTOR VIEWS QUESTIONNAIRE

Q3: I don’t.
Q4:
Q5:
Q6:

*instructor62
52011122442244443211303330132322
Q3:
Q4:
Q5: Such that the answer is not obvious. The aim of an exam or an assessment is to determine how much of the material taught a student has grasped.
Q6:

*instructor63
44001053444434511112222222222
Q3: Provide a series of easier questions where there is a high likelihood that students who have studied should be able to answer correctly
Q4:
Q5: I try to make them similar to the real answer, and at least sound plausible to a student who is not confident in the correct answer so that a guess is harder and thought is required to select correct answer
Q6:

*instructor64
246011044444355311112222212211
Q3: Immediate feedback for the students when preparing for the assessment if the appropriate tools are in place.
Q4:
Q5: I tend to do it in pairs - the correct answer and one that is close to it but is erroneous due to a commonly made assumption that is incorrect, and another pair which also look reasonable but have a fundamental flaw which allows them to be eliminated with a basic understanding of the concept behind the question.
Q6: The question has to be presented in a way in which the concepts behind the question are the focus, rather than understanding the context and/or detail of the question itself.

*instructor65
129011045244544532332322321111
Q3:
Q4:
Q5:
Q6:

*instructor66
77011045241435531211331223221221
Q3:

Q4:

Q5: One (or a number) of...

Making a prediction of what the most common problem solving/logic errors are likely to be, and working them through. “Mixing together” two concepts that might be muddled, and the answer based on that mixup. If I’m really stuck for that last distracter, it might be some random correct-“looking” answer, or a comedy option :)

Q6:
Appendix E

Guided Learning A

This appendix presents the caption of the questions in the *guided learning A*. It is available online at:
Invitation to Participate in a Research Project

Project Information Statement

Project Title: Debugging: A constructivist approach to learning programming

Investigators:

- Miss Shuhaida Mohamed Shuhidan, (PhD student, School of CS&IT, RMIT University)
  shuhaida.mohamedshuhidan@rmit.edu.au, (03) 99252738
- Dr Margaret Hamilton, (Senior Lecturer, School of CS&IT, RMIT University)
  margaret.hamilton@rmit.edu.au, (03) 99252939
- Dr Daryl D’Souza, (Senior Lecturer, School of CS&IT, RMIT University)
  daryl.dsouza@rmit.edu.au, (03) 99252947

Dear programming student,

You are invited to participate in a research project being conducted by RMIT University. This information sheet describes the project in straightforward language, or ‘plain English’. Please read this sheet carefully and be confident that you understand its contents before deciding whether to participate. If you have any questions about the project, please ask one of the investigators above.

Who is involved in this research project? Why is it being conducted?

This research project seeks to understand how people learn to program. Any student currently enrolled in Programming 1 can be involved in this project.

Why have you been approached?

We are conducting the research to find out how we can help you learn programming better. We are approaching you as a student enrolled in Programming 1, to be involved as a voluntary participant in this study. We plan to collect data at various stages during the semester, and all data will remain confidential.

What is the project about? What are the questions being addressed?

Programming is a difficult task, but essential for success in Computer Science study. We would like to understand what things make it difficult to learn programming initially. We plan to develop a Guided Learning Tool (GLT), specifically designed to aid students to learn programming. We have series of programming exercises; some of it based on the course materials and other sources. It will benefit you as there are explanations along while you are answering the questions.

If I agree to participate, what will I be required to do?

You will attempt to answer the programming task and fill in the feedback form.

What are the risks or disadvantages associated with participation?

Any participation will remain voluntary. We will ask for your student ID but this will not effect your grades. Thus, it can be stated that there is no risk to you for being involved in this particular study.

What are the benefits associated with participation?

Your participation will help us to further to improve the learning of programming. We aim to develop a tool to help novices in learning programming based on a fundamental understanding of where the novices are experiencing difficulty.

What will happen to the information I provide?

Any information you provide will be totally anonymous and will remain confidential. Only the members of the investigative team will see such data, which will be kept secure for a period of five years and then destroyed.

Because of the nature of data collection, we are not asking you to provide written informed consent of participation in the project.

Instead, we assume that you have given consent by clicking the button 'Yes, I agree to participate'

What are my rights as a participant?

Your rights as a participant of this project include:

- The right to withdraw their participation at any time, without prejudice
- The right to have any unprocessed data withdrawn and destroyed, provided it can be reliably identified, and provided that so doing does not
APPENDIX E. GUIDED LEARNING A

- increase any risk for you.
- The right to have any questions answered at any time.

Whom should I contact if I have any questions?

Any member of the investigative team listed at the beginning of this plain language statement may be contacted at any time. Any complaints about the conduct of this research project can be made to the Executive Officer, RMIT Human Research Ethics Committee, see http://www.rmit.edu.au/rd/hrec_complaints and the footer of this page.

Yours Sincerely,

- Ms Shuhaida Mohamed Shuhidan, MSc
- Dr Margaret Hamilton, PhD
- Dr Daryl DSouza, PhD

---

Any complaints about your participation in this project may be directed to the Executive Officer, RMIT Human Research Ethics Committee, Research & Innovation, RMIT, GPO Box 2476V, Melbourne, 3001.

Details of the complaints procedure are available from the above address or http://www.rmit.edu.au/rd/hrec_complaints
SECTION A: Detect and fix the program given

1. Please detect the error(s) by ticking the check box of the line contain error(s).

The program below has error(s). OUTPUT of the given Java Hello World Example would be:

Hello World

- 1. import java.io.*;
- 2. /*
- 3. Java Hello World example.
- 4. /
- 5. public class HelloWorld
- 6. {
- 7. public static void main(string [])
- 8.     {
- 9.         System.out.println("Hello World!")
- 10.     }
- 11. }

Total number of error(s):

2. Now, fix the program above:

```java
import java.io.*;
/*
Java Hello World example.
*/
public class HelloWorld
{
    public static void main(String [] args)
    {
        System.out.println("Hello World!")
    }
}
```

SECTION B: Answer the questions by referring to the program below.

1. import java.io.*;
2. // Java Hello World example.
3. 
4. public class HelloWorld
5. {
6. public static void main(String args[])
APPENDIX E. GUIDED LEARNING A

7. 
8. System.out.println("Hello ");
9. System.out.print( args[0] );
10. }
11. }

1. What does io stand for?
   ○ A. in out
   ○ B. input output
   ○ C. I do not know the answer for this question

2. What is the purpose of import java.io.*; ?
   ○ A. To make use of the method(s) of the library in other system
   ○ B. To import the input and output of the program to other system
   ○ C. I do not know the answer for this question

3. What is the purpose of /* …. */?
   ○ A. To comment out statement(s)
   ○ B. To terminate a statement
   ○ C. I do not know the answer for this question

4. What is the difference between /* ….. */ and // ?
   ○ A. /* ….. */ is used to comment out a line and // is used to comment out a block of statement
   ○ B. /* ….. */ is used to comment out a block of statement and // is used to comment out a line
   ○ C. I do not know the answer for this question

5. Choose the appropriate access control for a program based on its modifier

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>private</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>default</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>protected</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

6. What is the purpose of a semicolon (;) in a program?
   ○ A. To comment out a line
   ○ B. To terminate a statement
   ○ C. I do not know the answer for this question
APPENDIX E. GUIDED LEARNING A

7. What does args stand for?
   - A. arguments
   - B. agree
   - C. I do not know the answer for this question

SECTION C: Please mark as what you think best describes your opinion regarding the exercises above

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>a) These exercises help me learn programming</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>b) They are simply extra work for me</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>c) They are too hard</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>d) They help me to understand a basic structure of a program</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>e) They could help with any misunderstandings</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>f) They are too easy</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>g) They help me to recall the required information</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>h) I would like extra exercises like these to help me learn programming</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Yes, a lot</th>
<th>Yes, a little</th>
<th>Not at all</th>
</tr>
</thead>
<tbody>
<tr>
<td>i) I accessed the internet to help me answer the above questions</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>j) I used the course materials to help me answer the above questions</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>k) I used a text book to help me answer the above questions</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>l) I would like to add this question(s) to the exercise tested above:</td>
<td>[ ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m) My suggestion(s) for the Learning Guide will be:</td>
<td>[ ]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix F

Guided Learning B

Guided learning B is available at:
Guided Learning Tool

Hello shuhaida, Date: Sun, 15 Jan 2012 04:39:24

Learning objective: In this topic, you are exposed to basic syntax and the general idea of how a program works.

Instructions: Click on the hyperlinks in SECTION A for more questions in SECTION B to aid you to learn about the particular subtopic.

SECTION A

The following program print out: 

```
1. import java.io.*;
2. // Java Hello World example.
3. 4. public class HelloWorld
5.   6. public static void main(String args[]) 
7.   8. System.out.println("Hello World");
9.   10.}
```

SECTION B: Select the answer from the options given.

1. What does io stand for?
   - A. in out
   - B. input output
   - C. I do not know the answer for this question

Submit Next

SECTION C: Advance exercise for you to try:

Based on program in Section A, write a program that will ask the user to enter his/her name and print out "Hello" and the entered name.

The example of the print out is as below:

```
Please enter your name: Mr Happy
Hello Mr Happy
```

Figure F.1: Guided Learning B: Topic 1 of Week 1
Hello Shuhaida, Date: Sun, 15 Jan 2012 04:38:28

Learning objective: In this topic, you are exposed to operators and data types.

Instructions: Click on the hyperlinks in SECTION A for more questions in SECTION B to aid you to learn about the particular subtopic.

---

SECTION A
The following program print out:

```java
public static void main(String[] args) {
    int x = 30;
    int y = 20;
    int z;
    z = x + y;
    z = x - y;
    z = x * y;
    z = x / y;
    System.out.println("ADDITION: x + y = " + z);
    System.out.println("SUBTRACTION: x - y = " + z);
    System.out.println("MULTIPLICATION: x * y = " + z);
    System.out.println("DIVISION: x / y = " + z);
}
```

SECTION B: Select the answer from the options given.

What will be the value assigned to the variable x as a result of the following statement?

```
int x = 10 / 4 + 10 % 4 * 2;
```

- A. 0
- B. 4
- C. 6
- D. 8
- E. None of the above

---

SECTION C: Advance exercise for you to try:

Improve the programming code in SECTION A by creating a menu for user to choose on the operation, insert the digits and calculate it and give the result in two decimal points.

---

Figure F.2: Guided Learning B: Topic 2 of Week 2
Hello Shuhaida, Date: Sun, 15 Jan 2012 04:46:42

Learning objective: In this topic, you

Instructions: Click on the hyperlinks in SECTION A for more questions in SECTION B to aid you to learn about the particular subtopic.

**Question 3**

1. Based on the height/weight table, the following program will print out

<table>
<thead>
<tr>
<th>Height</th>
<th>Weight &lt; 65</th>
<th>Weight &gt;=65</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1.6</td>
<td>Normal</td>
<td>Over-weight</td>
</tr>
<tr>
<td>&gt;= 1.6</td>
<td>Under-weight</td>
<td>Normal</td>
</tr>
</tbody>
</table>

ConsoleReader console = new ConsoleReader(System.in);
System.out.print("Enter your weight in kg: ");
double weight = console.readDouble();
System.out.print("Enter your height in m: ");
double height = console.readDouble();

if (weight < 65)
if (height < 1.6)
System.out.println("Normal");
else
System.out.println("Under weight");
else
if (height < 1.6)
System.out.println("Over-weight");
else
System.out.println("Normal");

**SECTION B:** Select the answer from the options given.

If the value of the variable x is -10, 10 and 100 respectively then what is printed after the execution of the following code segment for each of the values?

21. if (x > 50 && x > 0)
System.out.println("Success!");
else
System.out.println("Failure!");

A. Failure!, Failure!, Success!
B. Failure!, Success!, Failure!
C. Failure!, Failure!, Failure!
D. Success!, Success!, Failure!
E. None of the above

**SECTION C:** Other exercise for you to try:

Complete the missing lines of the program below to print the gross salary for a weekly paid hourly rated employee. Hours in excess of 40 are paid at 1.5 times the normal rate. For example an employee working 30 hours at the rate of $10.0 per hour will be paid $300.00 while another employee working 60 hours at $10.0 per hour will be paid $700 (40*$10 + 20*1.5 * $10 )

```
double rate, hours, grossSalary;
ConsoleReader console = new ConsoleReader(System.in);
System.out.print("Enter hours worked : ");
hours = console.readDouble();
System.out.print("Enter hourly rate: ");
rate = console.readDouble();
```

System.out.println("Gross salary is " + grossSalary);

---

*Figure F.3: Guided Learning B: Topic 3 of Week 3*
Hello shuhaida, Date: Sun, 15 Jan 2012 04:39:50

Learning objective: In this topic, you .
Instructions: Click on the hyperlinks in SECTION A for more questions in SECTION B to aid you to learn about the particular subtopic.

**Question 4**
The following program print out :

```
public static void main(String[] args) {
    for(int a=5;a>=1;a--){
        System.out.println("*");
        for(int b=1;b<=a;b++){
            System.out.print("*");
        }
    }
}
```

**SECTION B: Select the answer from the options given.**

What is the output of the following code?
```
int b = 2;
String a = "BOB";
if(a.indexOf('O') == b){
a += "Bill";
b = 3;
}
for(int i = 1; i < b; i++){
    System.out.println(a);
}
```

- A. BOBBOB
- B. BOBBillBOBBillBOBBill
- C. BOB
- D. BOBBillBOBBill
- E. BOB

**SECTION C: Advance exercise for you to try:**

With the aid of the program in Section A, Use nested for loops to print the pattern of stars in the form below:

```
*
**
***
****
*****
*
**
***
****
*****
```

**Figure F.4: Guided Learning B: Topic 4 of Week 4**
Guided Learning Tool

<table>
<thead>
<tr>
<th>Basic Syntax</th>
<th>Operators</th>
<th>Selection</th>
<th>Iteration</th>
<th>Other</th>
<th>Instruction</th>
<th>Log Out</th>
</tr>
</thead>
</table>

Other helpful links

- [How to install Eclipse](#) (Credit to University of Maryland)
- [Putty](#)

Figure F.5: Guided Learning B: Other Useful Information
Welcome to the Guided Learning Tool which provides exercises for learning in Java.

These exercises are intended to help you understand important concepts in introductory programming. They range from simple programs which may require you to write some basic code to more difficult exercises which require some chunks of code.

SECTION A provides a sample of a working program for you to study. There are links for new terms to remember and understand. Click on the links and a series of multiple choice questions will appear in SECTION B.

Select the best answer and press the Submit button to check the answer. If you get them wrong, these may highlight a gap in understanding the concepts. If you get the answer wrong or you do not know the answer, please click on the "Do not know" option to direct you to the learning materials which may explain the concepts in detail. Click on the Next link in SECTION B to go through to more challenging questions.

SECTION C provides a more advanced question on the same topic for you to apply your knowledge on the subject matter. You will need to run your answer to this question in any IDE (Integrated Development Environment) such as BlueJ, Eclipse or in command-line.

Figure 1 explains the logic behind this tool represented in terms of the Bloom's Taxonomy levels of "Remember", "Understand" and "Apply".

Figure 1: Using Bloom's Taxonomy to Classify Questions in Guided Learning Tool
Guided Learning Tool

Please take a moment to fill in this short survey

<table>
<thead>
<tr>
<th>Instruction: Please mark as what you think best describes your opinion regarding the Guided Learning Tool</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) These exercises help me learn programming</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>2) They are simply extra work for me</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>3) They are too hard</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>4) They help me to understand a basic structure of a program</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>5) They could help with any misunderstandings</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>6) They are too easy</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>7) They help me to recall the required information</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>8) I would like extra exercises like these to help me learn programming</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>9) I accessed the internet to help me answer the questions</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>10) I used the course materials to help me answer the questions</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>11) I used a text book to help me answer the questions</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>12) My suggestion(s) for the Guided Learning Tool will be:</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Thank you for filling in the survey, it is very much appreciated

If you have any further question, email us at: shuhaida.mohamedshuhidan@rmit.edu.au

Figure F.7: Guided Learning B: Survey


BIBLIOGRAPHY


BIBLIOGRAPHY


268


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


