INCORPORATING VISUAL AND ANIMATION TEACHING TOOLS IN COMPUTER PROGRAMMING CLASSES FOR EFFECTIVE TEACHING AND LEARNING

MADASARI BTE ANSARI

A Master's Project submitted in partial fulfilment
of the requirements for the degree of
Master of Information Technology

Faculty of Information Technology and Multimedia Communications
Open University Malaysia

2011

ABSTRACT

The problems in teaching and learning programming techniques prevail all over the world. Learning programming logic without utilizing any visualization material is difficult. One possible solution to overcome this problem is by using program animation. The proposed program animation software used in teaching novices learning introductory programming is JELIOT 3. The aim of this project is to investigate the effectiveness of teaching programming classes for beginners using programming tool JELIOT 3 that incorporate visualization and animation, specifically the outcome of students' performances before and after using JELIOT 3. The study population is novice students with no significant programming experience. They were drawn from an on campus section of second semester Foundation in Business Information Technology introductory programming course at INTI International College, Subang Jaya. The class size when this study was conducted was only 9 students. The 9 students were divided into two groups at random. The first group (POST-JELIOT) received standard classroom lectures then followed by JELIOT 2. The second group (PRE-JELIOT) received the JELIOT 3 lesson and then the traditional classroom lecture. From the findings, there was a significant difference in performance when comparing between students who are first taught JELIOT 3 in the overall assessments on the two programming tests and on the two array model assessments. The implications then are that a visual teaching and animated illustration to programming is an effective method for teaching programming. Instruction in computer programming must make use of such visualization to develop good mental visualization of programming.

ABSTRAK

KEGUNAAN BAHAN DAN CIRI GAMBARAN DAN ANIMASI DALAM MENGAJAR KELAS PERISIAN KOMPUTER UNTUK MENGAJAR DAN BELAJAR DENGAN BERKESAN

Masaalah mengajar dan mempelajari teknik perisian komputer berleluasa. Mempelajari perisian computer tanpa menggunakan sebarang animasi atau bahan-bahan yang mewujudkan gambaran mental kemungkinan menghalang penuntut untuk mempelajarinya dengan baik. Justeru itu, kajian ini telah menggunakan JELIOT 3 untuk mengajar penuntut yang mangambil kursus pengenalan perisian komputer. Tujuan kajian ini adalah untuk mengetahui hasil mengajar kelas perisian komputer dengan menggunakan JELIOT 3 yang dimasukkan dengan ciri-ciri animasi dan gambaran mental, terutama terhadap pencapain penuntut sebelum dan selepas menggunakan JELIOT 3. Kajian sampel adalah terdiri daripada 9 penuntut semester 2 yang mengambil kursus pengenalan perisian computer di Kolej Antarabangsa INTI, Subang Jaya. 9 penuntut ini di bahagikan kepada dua kumpulan tanpa kaedah. Kumpulan pertama iaitu, POST-JELIOT memperolehi pengajian tradisi di dalam kelas sebelum menggunakan JELIOT 3 dan kumpulan kedua iaitu, PRE-JELIOT memperolehi pengajian menggunakan JELIOT 3 sebelum pengajian tradisi di dalam kelas. Daripada kajian tersebut, terdapat kesan yang memperangsangkan terhadap pencapaian keseluruhannya apabila di bandingkan penuntut yang memperolehi pengajian JELIOT 3 dahulu sebelum pengajian tradisi. Implikasi kajian ini adalah bahawa menggunakan animasi dan menggunakan gambaran mental dapat mewujudkan cara mengajar yang berkesan. Peraturan dalam perisian computer mesti menggunakan bahan-bahan atau ciri-ciri gambaran mental untuk membina gambaran mental yang baik terhadap pengajian perisian komputer.

TABLE OF CONTENTS

CHAPTER ONE: INTRODUCTION

1.1 Background to the study..……..……..1
1.2 Problem Statement ……...2
1.3 Objectives of the Study……....4
1.4 Significance of the Study.....…….. 5
1.5 Limitation of the Study.....……....6

CHAPTER TWO: REVIEW OF LITERATURE

2.1 Previous Studies.....…….....7
2.1.1 Software Visualization.....7
2.2 Theoretical Framework.....…… 14

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Methodology..……..15
3.2 Data Collection Methods……..…….. 16
3.3 Research Population…….....……..17
3.4 Procedures…….....….18
3.5 Post-Test And Pre-Test.....…….…….....19
3.6 Jeliot 3 Lesson…….....….21
3.7 Measures.....……..…… 23

CHAPTER FOUR: RESULTS AND DATA ANALYSIS

4.1 Introduction……..25
4.2 Array Mental Models Assessments…… 32
4.2.1 Evaluation Methods……..…….32
4.2.2 Mental Model Assessment Results...……..33
4.3 Jeliot Lesson Observation Results…….….. 34
4.4 Standard Classroom Observation Results…….... 34
4.5 Array Model Assessment Discussion…… 35
4.6 Conclusion……36

CHAPTER FIVE: DISCUSSION AND CONCLUSION

5.1 Summary of Main Findings…....…… 37
5.2 Discussion of Findings and How It Is Related To
Teaching Programming……..……...38
5.2.1 Limitations……..…… 42

5.2.2 Implications For Jeliot Lessons……..... 42
5.3 Directions For Future Research……... 43

REFERENCES….....45

APPENDICES
APPENDIX A: Post and Pre Jeliot Class Sessions……..…….50
APPENDIX B: Pre-Array Mental Model Assessment…….....52
APPENDIX C: Post-Array Mental Model Assessment.....53
APPENDIX D: Programming Test 1 Answers……...54
APPENDIX E: Programming Test 2 Answers……....60
APPENDIX F: Jeliot Lessons…….....66

LIST OF TABLES

Table 1: Independent Samples Test For First Programming Test.....….....27

Table 2: Group Statistics for First Programming Test.....28

Table 3: Independent Samples Test for Second Programming Test.....30

Table 4: Group Statistics for Second Programming Test.....31

Table 5: Independent Samples Test for Array Model Assessments.....32

Table 6: Group Statistics for Array Model Assessments.....32

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND TO THE STUDY

Most students are still being introduced to programming by reading texts, listening to lectures and writing programs. Is this an effective method for teaching most students and does it reflect the changes in technology and teaching methods that have been developed? Computer systems have improved and programming languages have become more sophisticated. This increases the demands for creativity on novice learners.

Students who receive assistance in developing a proper mental model of programming will have a better understanding of a program. It is so essential today to know how to program effectively. Software engineering is a field that is highly demanded in Malaysia. Most companies are depending on computerised system for their operational processes. With lack of logical thinking, analytical mind and accurate mental model in programming, these programmers may have trouble creating an effective mental model of how a computer system works and hence lead to poor quality of system.

The purpose of this study was to investigate how JELIOT 3, which is a visualized and animated programming teaching tool, could effectively help students to develop conceptual models related to programming. Novice programming students encounter many challenges when learning a programming language. Their learning is confounded

by poor mental models related to programming and the lack of a conceptual framework on which to build their knowledge. The introduction of a good conceptual model of programming concepts could improve students' understanding. This study investigates two groups of novice programming students to see if JELIOT 3 improves the mental model of students and their understanding of programming concepts using arrays.

1.2 PROBLEM STATEMENT

The problems in teaching and learning programming techniques prevail all over the world. Learning programming logic without utilizing any visualization material is really tough [Robins et al; 2003; Lahtinen et al; 2005]. To improve the teaching and learning environment, development of novel and appropriate learning aids are therefore highly relevant [Naps et al; 2003]. Numerous visualization tools are already available for beginners in programming and these visualizations are capable of providing a more conducive learning environment if they are used effectively.

Learning programming is a complex task for many novice computer science students at university level. Many students tend to think that learning programming is just about learning some programming language syntax. However, this is normally the easiest part. The main important problem for many students is their low ability to develop an algorithm that solves a given problem efficiently. The application of basic concepts, like control structures or the design of simple algorithms can be difficult obstacles for many students. These difficulties are not dependent on the programming

language used or even on the programming paradigm. Some authors identified reasons for these difficulties [Jenkins T, 2002], such as:

- The need for good competences on problem solving;
- The importance of acquiring programming skills do have lots of contributions to the employability and marketability of oneself;
- Students are used to courses that depend mostly on theoretical knowledge and memorization, but basic programming learning needs a more practical approach, based mostly on problem solving activities;
- Traditional teaching, often based on lectures and specific programming language syntaxes, often fails to motivate students to get involved in meaningful programming activities;
- Programs have a dynamic nature, but most learning materials have a static format which makes difficult to analyze program's dynamic behavior and
- Students learning needs are often very different within the same course.

The dynamic nature of programs suggests that their operations and interactions can be well described by visual representations. Hence, the utilization of visualization, animation and simulation tools seems a natural option. This type of tools is based on the mapping of computational representations through visual representations that allow the concentration of programming abstraction. This can help students understand the program's behaviour and also facilitate communication and collaboration between the students and between them and the teachers.

The main goal of this project is to promote best practices for teaching and learning programming effectively using JELIOT 3. Students are also given an exposure

to using JELIOT 3 which is available and be downloaded for free from http://www.cs.joensuu.fi/jeliot/.

1.3 OBJECTIVES OF THE STUDY

Automated learning is an exciting new dimension to explore and promises much. As technology progresses, a more effective environment for learning will evolve. Adaptations and improvements will have to be made to fine-tune the facilities available to what can be offered as features of an automated learning environment.

The aim of this project is to investigate the effectiveness of teaching programming classes for beginners using programming tool that is JELIOT 3 that incorporate visualization and animation.

Specifically, this project seeks to achieve the following objectives:

1) To introduce JELIOT 3 to programming students.
2) To determine the pre performance of the students prior to the introduction of JELIOT 3.
3) To evaluate the post performance of the students after using JELIOT 3.
4) To evaluate the students' array mental model.
5) To recommend the methods of utilizing JELIOT 3 in teaching programming classes.

1.4 SIGNIFICANCE OF THE STUDY

One of the distinctive contributions of this project is to enhance understanding of learning programming in higher learning institutions to promote solid foundations and hence create better chances of employability among computer science students in Malaysia. A weakness in our current knowledge in programming subjects is that we do not fully understand and fully utilize, in terms of logical design of the programming modules and what works best in different situations. Teaching and learning programming languages is different and complex. By using appropriate tools, the teaching and learning of programming languages can work better and in a more interesting and captivating manner. In order to achieve an effective teaching methods, the animation tools can enforce the quality of programming teaching offered in a traditional university environment.

The results of this project will provide insights that will benefit students, by discovering the important criteria, requirements and tools that should be included and implemented in a learning a programming subject and how these requirements and tools can help to enhance learning and teaching.

This project identifies the effectiveness of using animation teaching tools in the teaching of programming subjects. To know how to present in effective ways enable the lecturers to maintain high levels of communications between the students and the lecturers to promote the desired levels of knowledge.

1.5 LIMITATIONS OF THE STUDY

This project and its output are intended to be an introduction for the foundation program students majoring in information technology at INTI College, Subang Jaya. These students are learning Java programming as an introductory for programming course. They have only 14 weeks of study week and JELIOT 3 will be introduced to them at the beginning of the class. Normally the class is small between 6 and 10 students. Thus, the sample data that was used depend on the number of students that enrolled in this program for a semester. This number is small and is out of the researcher's ability to increase the number.

Even though Java programming, which is Object Oriented programming software is introduced to the students, the course structure emphasized on the structured programming aspect. Structured programming is often associated with a top down approach to design. Top down approach to problem solving is the technique of breaking problems into parts and solving each part. Once each part is fully understood, the solution to the overall problem can be readily developed. Structured programming embodies a disciplined approach to writing clear code that is easy to understand, test, maintain and modify. Structured programs are often composed of simple, hierarchical program flow structures. These are sequence, selection and repetition. The areas to be focused are therefore only on basic structure of programming, use of variables and its data types, control structures such as sequential, repetition and selections, the use of one and two dimensional array, searching and sorting and finally the use of local and global variables in a program.

CHAPTER TWO

REVIEW OF LITERATURE

2.1 PREVIOUS STUDIES

2.1.1 SOFTWARE VISUALIZATION

The difficulties of learning to program are well known. One solution that can be used is the use of program animation. The proposed program animation system to be used for teaching novice students learning introductory programming is JELIOT 3.

JELIOT 3 pioneered the use of self animation, where the animations are produced automatically from the source code of programs without any effort whatsoever on the part of the teachers or the students. This capability is important for the students as well as teachers; it is well known that educational technology like visualization will have a difficult time being accepted if much effort needs to be invested in order to use it. With JELIOT 3, this effort is negligible.

The JELIOT family is a collection of program visualization environments that has been developed over the past ten years (Ben-Ari et al; 2002). The latest version JELIOT 3 was developed at University of Joensuu (Moreno et al; 2004) and designed to ease teaching programming concepts top novices and aid in program comprehension and debugging. Its predecessor, Jeliot 2000, has been successfully used in programming

concepts and explain unseen situation (Ben-Bassat Levy7 et al; 2003). This might be due to the fact that the visual and concrete model of JELIOT helps the learner to build a viable mental model of the program execution and use it to verbalize the execution (Mayer, 1981).

A survey of work on software visualization can be found in Stasko, Dominique, Bewon and Price (1998). This work is intimately connected with more general work on multimedia learning. Mayer (1997) consistently found that visualization needs to be accompanied by explanations to be effective. Similarly, Petre and Green (1993) found that graphics are not self explanatory; therefore, it does not follow that the use of these tools in a classroom will automatically improve learning.

The first research specifically directed at investigating learning of programming using visualization was done by John Stasko. Algorithm animation was used to teach a complicated algorithm to graduate students in computer science. The results however were disappointing. In a post test, the group using animation did not perform better than those that did not. His later work demonstrated a slight improvement (statistically insignificant) in student learning when the students were asked to predict the outcome. Hundhausen, Douglas and Statsko (2002), suggest that it is not enough to have a good visualization tool as its effectiveness depends on how it is used. They defined the term system roulette for the situation where visualization systems are unused system, because their developers did not deal with the real needs of their potential users.

An ITiCSE Working Group report summarized the advantages and disadvantages of using visualization tool (Naps, Robling, Almstrum et al, 2002). The advantages include: assist students understand topics in computer science; make teaching more

enjoyable; facilitate discussion in class. The disadvantages include: duration to install software tools; effort required to introduce the tools into teaching practice; lack of information about the benefits for learning.

During the years 1998 – 2001, a study was conducted to evaluate the effect that Jeliot has on students when it was integrated into a yearlong course in an introductory course, Fundamental of Computer Science (Ben-Bassat Levy, Ben-Ari, Uronen, 2003). There were two classes, one received an extra hour of instruction using Jeliot, while the control group received an extra hour of instruction using the same content but without Jeliot. There were differences between the grades of the group who used Jeliot and the control. The differences were directly proportional to the amount of animation used. The Jeliot group showed better improvement than the control group. It was found that usage of Jeliot over a long period improves both understanding and transfer of knowledge. Most importantly, it primarily helps the average students who might otherwise get a low grade, by enabling them to build concrete models of how computer works inside.

Earlier laboratory studies of software visualization were not able to demonstrate consistent improvement of learning, but claimed that the use of visualization systems improved characteristics such as motivation. Ebel and Ben-Ari (2006) attempted to prove this experimentally. Attention was considered as a prerequisite for effective learning and investigated possibility of Jeliot improving attention. Ebel and Ben-Ari found that there were no disruptive episodes. Nevertheless, it is not be expected that the continuous use of Jeliot would prevent disruptive behavior, but the results do justify using such educational software occasionally improve learning directly. The use of

animation systems is not widespread even though they are effective learning tool (Hundhausenm, Douglas and Stasko, 2002).

The advantages of using visualization in learning have been proven (Lawrence et al. 1994). Computer based visualization is an effective support for modeling (Crapo et al. 2000). The same conclusion was made by Meisalo et al (1997). Visualization leads students' attention and helps them to understand the connection between concepts and text (Mayer 1997). Students learn more easily through observation and analysis of the visualization from different inputs (Anderson and Naps 2001). Visualization maintains student's power of concentration, clarifies complicated concepts, automates examples and presentations and increases communication between teachers and students (Khuri 2001). Visualization should be used when teaching materials that are to be difficult (Torvinen 2001). Kehoe et al. (1999) found out that visualization makes students more motivated and pleased to learn programming. Sutinen et al. (1997) also discovered that students are more motivated when visualization is used. The students who are using visualization produce source code and written documentation of higher quality (Markkanen et al. 1998). Their development is more effective and the vocabulary of the terms of programming is composed faster (Ben- Bassat Levy et al. 2003).

The criticism against visualization concerns the effectiveness of animation. (Petre, 1995) found out that novice students have insufficient skills to understand graphical displays and lack the overall view of information. Graphical displays are not self-explanatory; instead students have to be taught to understand them (Petre and Green, 1993). According to Hundhausen et al. (2002) in visualization a picture is not worth of

thousand words. Even novice students have also difficulties understanding each graphical element images in the algorithm (Stasko et al. 1993).

The good quality visualization helps to learn but the focus is on the form of activity and learning exercises used. The students' opinions vary depending on the strength of students. Visualization is too confusing for weak students but is deemed unnecessary for strong students (Ben-Bassat Levy et al.2003). Petre et al. (1998) disagree by saying that the professional users use visualization to converse with equals and as some kind of tool of thinking whereas novice use visualization to see how algorithms work, for a new way of imaging the problem and to get a glimpse of professional world of ideas. That is the reason why visualization should offer different kinds of views for users at different levels.

Shahjalal University of Science and Technology, Bangladesh has become an active partner in the global learning program through the Codewitz Asialink project (Codewitz 2004). The goal is to plan, produce and evaluate unique illustrations, animations and visualization aids for students and teachers of computer programming. Even if the use of visualization has increased in introductory programming, the visualizations are still integrated in the course content. Four-party collaborative team in partnership with Tampere Polytechnic of Findland developed a considerable number of Learning Objects (LO), which are significantly different from some of the existing learning aids in terms of pedagogic principles incorporated into their design. The LOs were then integrated with the courseware in the partner's institutes. The results of the program showed that students who used LO's performed better than students who did not use them. Within the Codewitz network, the LOs are defined as visual tools for learning

that are browser capable, stand-alone, reusable and not linked to any other LOs or resource and are focused on specific learning goal. The LOs are based on visualizing programming logics and suitable for exercising.

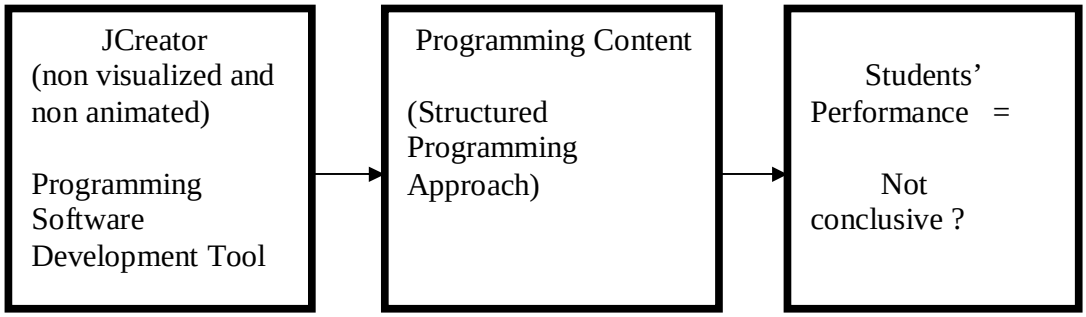
The idea of LOs resembles a debugger which shows step-by-step program execution in both forward and backward directions. The program code is highlighted in each meaningful step of execution. Every step of program execution is visualized at the same time in the memory area, Visualization Windows. An Information Window is also included for explaining the statement-sensitive information. The memory area is the only part where the layout can be changed according to the subject-matter of the underlying program. This change appears for example, in case of array when the structure of the array is visualized. An LO based on a first-day C program for adding two integers. Declaration of three variables, b and c is depicted by visualizing three memory boxes in the visualization window, which is then followed by stepwise appropriate visualization of value assignment to a, b and c. Another LO depicting a relatively complex problem using for loop derives substring from a string starting from a given position and length. Visualization of the loop variables together with the process of copying characters to a second array provides a clear understanding of the logical flow of looping structure. Different set of input can also be exercised by moving the flow of execution back and forth, which additionally gives interactive learning environment to the students.

A study was conducted in Shahjalal University of Science and Technology for Computer Engineering students who were majors of two programming courses. The course “Structured Programming Language” conducted in first semester using C language typically covers the uses of selection logic, loops, arrays, functions, structures

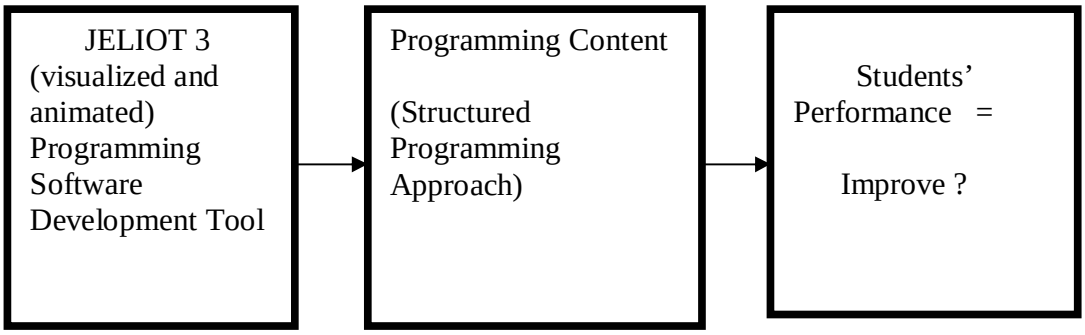
and files. The course consisted of lectures (2 hours/week) and lab exercises (6 hours / week). Students do not have prior programming experience since the course was offered in the first semester. 105 students (of Session 2006) were grouped into two sections (Section A: 53 students and Section B: 52 students) in order to evaluate the performance of learning objects. Students having odd and even registration number were grouped into Section A and Section B, respectively and taught by the same instructor. The students of Section A were taught with the aid of LOs while the students of section B were delivered verbal lectures using traditional white board. The program visualization LOs was also available for the students of Section A when studying at home. The students found the aids interesting as learning tools. The final grades obtained by the students of Section A and B clearly demonstrate that students using visual aids performed quite better than those without using the LOs.

An important insight in the figure is that a large number (30) of Section B students (without using visual aids) obtained grades between 60 - 69 and 70 – 79, whereas 27 of the Section A students obtained grades between 70 – 79 and 80 – 89. This shift in grades illustrated a clear improvement of students' performance using LOs.

2.2 THEORETICAL FRAMEWORK



The above diagram illustrates the conventional teaching of Java using JCreator, which is a non visualized and non animated teaching tool on teaching structured programming to novice programming students. The results of students' performance are questionable and not conclusive.



The above diagram illustrates what this study is about. To find out if using JELIOT 3, which is a visualised and animated teaching tool will improve students' performance comparatively.

INTI
International College Subang
LAUREATE INTERNATIONAL UNIVERSITIES

3, Jalan SS 15/8
47500 Subang Jaya
Selangor, Malaysia
T +603 - 5623 2800
F +603 - 5636 7459
www.newinti.edu.my

Center of Pre-University Study

Foundation in Business and IT
CSC1211 PROGRAMMING TECHNIQUES
Semester : 2

TEST 1

Instructions:

1. **The student must answer all the questions. Any answer that did not conform to the question will not be given any marks or deducted.**
2. **Any dishonesty case will not be tolerated.**

STUDENT'S PARTICULAR:

Student's name	
Student I.D.	
Signature	

MARKING :

Section	**A**	**B**					**TOTAL**

MARKS							

Section A

Instructions: This section consists of **20** questions. Answer **ALL** questions. All questions carry equal marks.

1) In the computer, a variable is :
   (A) A single line of code
   (B) The answer to a math problem
   (C) A storage location for data
   (D) One trip through a loop

2) New variables are created using :
   (A) A declaration statement
   (B) An int statement
   (C) An assignment statement
   (D) A Name statement

3) The correct order for coding most programs is :
   (A) GIGO
   (B) FISH
   (C) LIFO
   (D) IPO

4) How many times will this loop repeat?

```
int count=0;

while (count < 3)
{
    count = count + 1;
    System.out.println(count);
}
```

   (A) 0
   (B) 1
   (C) 3
   (D) infinitely

5) The variable you would use to find the running total for a series of numbers is called :
   (A) A counter
   (B) An accumulator

(C) A string
(D) A loop

6) In the variable testscore[3], the 3 is the :
   (A) Value stored in the variable
   (B) Subscript
   (C) Size of the array
   (D) Type of array variable

7) An array is :
   (A) A single variable
   (B) Multiple variables with the same name
   (C) A method of reading values from a file into the computer
   (D) A series of steps repeated by the computer

8) What coding change would you make so you could have more items in an array?
   (A) You would need to change the number of times the loop repeats
   (B) You would need to change the number of the array variables
   (C) You would need to change the type of the array variables
   (D) You would need to add a line of code for each new variable

9) Variables in an array can be distinguished from one another by their :
   (A) Name
   (B) Assigned value
   (C) Subscript
   (D) type

10) A programming loop is similar to :
   (A) Steps in recipe
   (B) Verses in a song
   (C) Driving directions
   (D) Classrooms in a building

11) In which of the following is intTotal an accumulator?
   (A) for (intNum = 0 ; intNum <=5;intNum++)
       intTotal = intTotal + intPay;
   (B) for (inNum = 0; intNum <=5; intNum ++)
       intTotal = intTotal + 1;
   (C) if (intTotal < 5)
       intTotal = intTotal + intPay
   (D) if (intTotal < 5)
       intTotal = intTotal + 1

12) Which of the following will assign 1998 to the third variable in the intDate array?
   (A) intDate = 1998;