DE LA SALLE UNIVERSITY - MANILA

PRO_LOGIC: COURSE REVIEWER

A Term Project

Presented to Mrs. Armie E. Pakzad

In Partial Fulfillment of the

Requirements for the Course Programming Logic and Design (PROLOGI)

By

BELANO, Paolo (MI) - (signature/initials)

HERNANDEZ, Miro Manuel Lugos - (signature/initials)

LIM, Ella Janelle U. - (signature/initials)

EQ1
T-TH 0800 - 0900

May 29, 2021

Table of Contents

List of Figures

I. Introduction

This section of the paper delves into the rationale of the project.

A. Background of the Study

This project was created in compliance with the term project requirement for the course Programming Logic and Design Lecture (PROLOGI). The scope of the possible topics simply stated that the program the students create must be, in some way, connected to engineering. The students of this group decided to develop a program that could be used to review the concepts that one may encounter in PROLOGI.

Since PROLOGI is an introductory course to the world of programming, the students recognize its lessons as the most fundamental principles of programming and felt that it would prove useful to be able to look back on such concepts, in the future, if need be. In addition to the ability of relearning and reinforcing, the students also felt it necessary to include a test option that would allow users to numerically assess their current proficiency in PROLOGI. Beyond simply an assessment of proficiency, the students also hope to promote learning through the test feature. As reported by Chou & Powers (2019), testing improves memorization and conceptual learning; albeit, their study was based around categorical learning and Chinese characters.

On another note, Gao & Hargis (2010) looked into the effectiveness of computer programs in learning. They found that there needs to be a balance between hands-on experience and teacher guidance. Since PROLOGI is merely a lecture course, the process of creating the output of this project would aptly serve as that hands-on experience between the lectures and the book learning the students have been doing.

In the era of online learning, some studying practices students used to enjoy are no longer viable. As such, the students of this group seek to create a program that could add to their learning arsenal one more alternative method of learning in hopes of ever-so-slightly motivating them to continue to do their utmost in their studies in the "new normal".

B. Problem Statement

As the laboratory and lecture classes were formulated to work hand-in-hand to deepen a learner's understanding of the concepts they are presented with. In De La Salle University, they are taught separately by different professors, sometimes, lessons do not necessarily align. While the ideas taught in counterpart classes remain fundamentally the same, to the untrained eye, they could be perceived as different ideas. The subsequent confusion, realization, and, possibly, discouragement could hinder, or worse, discourage, further learning.

C. Objectives

1. General Objective

The general objective of this project is the creation of a program that would aid in the reviewing of lessons discussed in PROLOGI.

2. Specific Objectives

The specific objectives of this project are to:

i. Create a program that would allow users to numerically assess their proficiency in the lessons from PROLOGI;

ii. Create a program that would make it convenient for individuals who want to look back on the fundamentals taught in PROLOGI to do so; and,

iii. Grant users a learning option outside of the traditional definition of studying.

D. Significance of the Project

Through this project, the students seek to develop a C++ program for reviewing the concepts in Programming Logic and Design Lecture (PROLOGI). Recognizing the importance of the most fundamental concepts in programming logic as they look towards more advanced programming courses, they felt that it would be useful to have a readily accessible means to look back on the basics further down the line. Beyond just the project output, the undertaking also grants the students the opportunity to practice their skills in both the lecture and the laboratory sense.

II. Review of Related Literature

This section looks into previous published studies involving the use of computer programs in learning. The studies below show their effects on students' performance and motivation.

1. In 2018, Jaberi studied the effects of studying using computer programs and applications by undergraduate students on their motivation and academic performance. He examined a sample of 500 students from Petra University to conduct this study. He found out a positive relationship with the amount of usage of computer programs and applications in students' motivation and academic performance. Since this type of learning is learner-centric, computer programs help students more effectively as compared to traditional learning methods. He highlights the importance of using computers as a way to teach students especially in the technological progressive world.

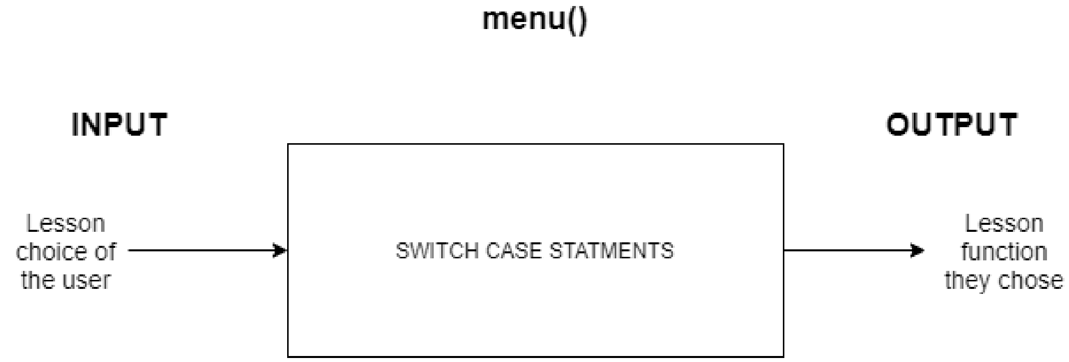
2. Waraich (2004) used a narrative-based framework to motivate students who had difficulty in studying computer architecture of binary numbers and logic gates. He developed a prototype of Interactive Learning Environment (ILE) that attempts to teach binary arithmetic and conversions through computer games and other multimedia platforms. ILE is student-oriented by guiding the students step-by-step through the exercise. He added a program called Binary Arithmetic Tutor (BAT) that uses fantasy narrative in context with the number system and other computer-related information. The results show that narrative-based approach (ILE) and goal-directed exercises (BAT) help in students' motivation and understanding of computer architecture.
3. Kempthorne & Steele (2014) compared three different approaches - lectures, group activities, and 'the game method' - to teaching binary, hex, and decimal conversions. They found that a game-based approach when teaching numeral system conversions is most effective for students without prior experience to the subject. However, they noted that his sample size was small and that similar studies need to be conducted to strengthen the reliability of their findings.
4. Dewhurst, et al. (2000) studied if independent learning through computers can be an alternative for lectures. They used computer-based learning (CBL) in specific modules and compared the students who used the CBL system to the traditional system. The results showed CBL was effective as shown by the performance of the students who use the CBL materials compared to those who didn't. The students gave general positive feedback on CBL as a method for learning.
5. Gao and Hargis (2010) studied the effectiveness of using technology in learning computer science. The researcher compared three methods of teaching: traditional teaching, active learning with less teacher-guidance, and active learning with more teacher-guidance. Active learning is defined as the students learning on their own using computer programs. After collecting the data using qualitative testimonials and test evaluation they found out that active learning with more teacher-guidance is the most effective. They emphasize that the combination of lectures and independent learning through computers gives students the needed information and hands-on experience.

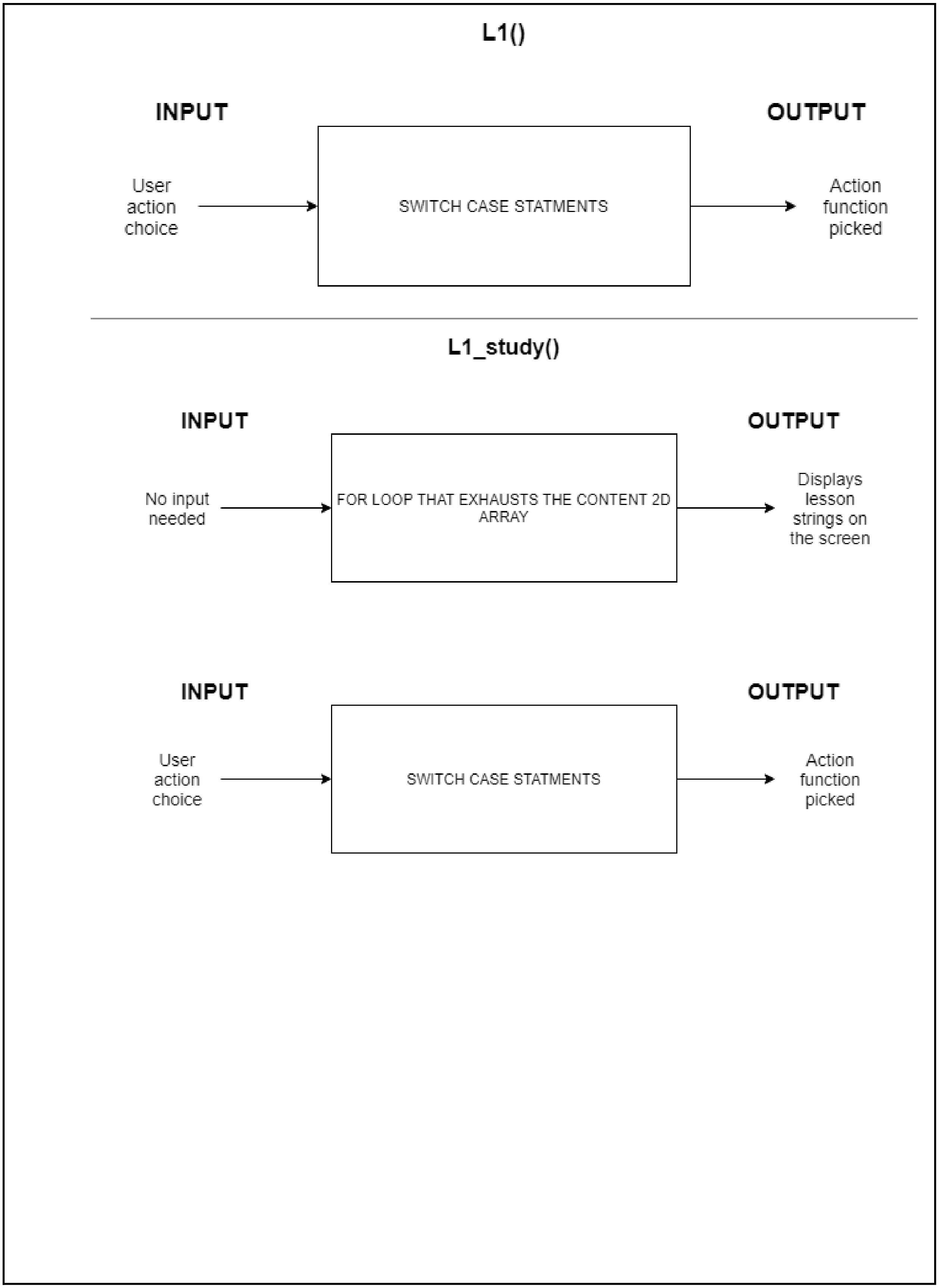
6. Keengwe et al (2008) published a paper about the different research about the importance of computers to support learning. In their synthesis, they emphasize how computers can be effective learning tools for students. The researchers also suggested that universities and academic institutions incorporate computers in their system especially for independent learning.
7. Harris et al (2016) examined the effects of one on one technology in students' academic achievement and motivation. Using 4th graders from an elementary school in Central Illinois, the researchers divide the sample into two: implementation classroom and traditional classroom. The implementation classroom used one on one technology to teach the students or independent learning using computer programs while traditional classroom used lectures. In the data, the researchers found out that the students from the implementation classroom scored higher during the test evaluation. Although they did not find correlation between technology and student's motivation or rather it is not a factor.

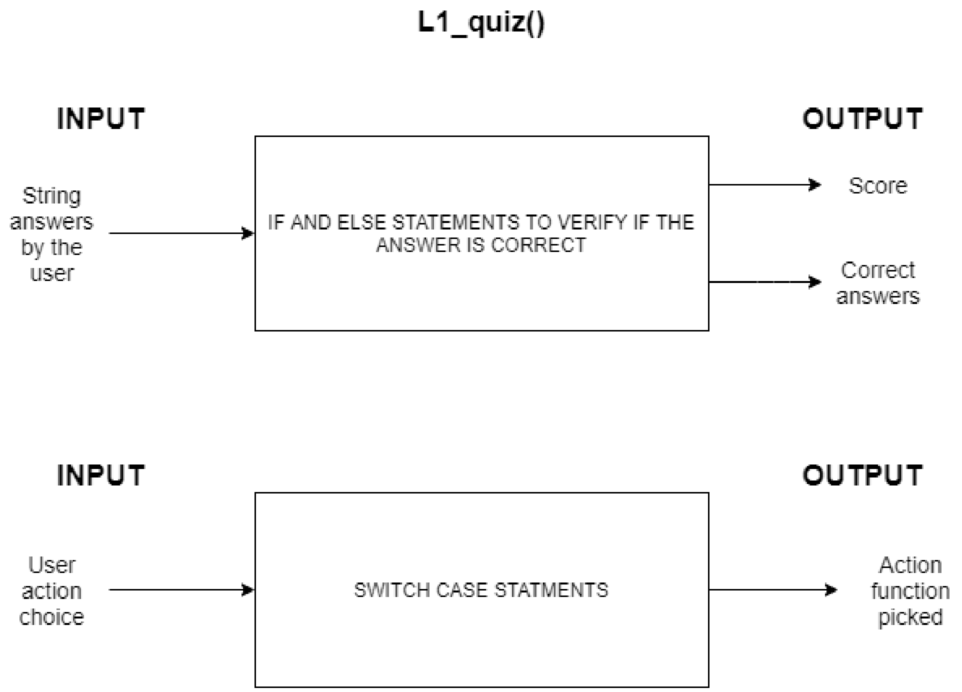
III. Methodology

This section of the paper is a narration of the conceptualization of the program logic used in source code of the final output.

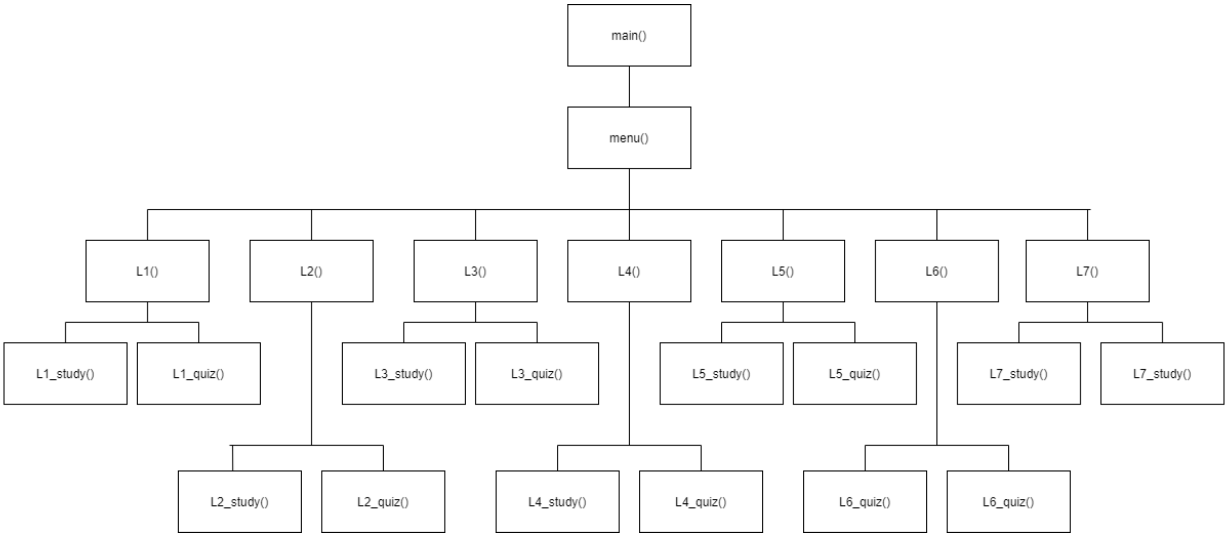
A. Conceptual Framework - IPO Chart




L1()
INPUT
OUTPUT
User action choice
SWITCH CASE STATMENTS
Action function picked
L1_study()
INPUT
OUTPUT
No input needed
FOR LOOP THAT EXHAUSTS THE CONTENT 2D ARRAY
Displays lesson strings on the screen
INPUT
OUTPUT
User action choice
SWITCH CASE STATMENTS
Action function picked

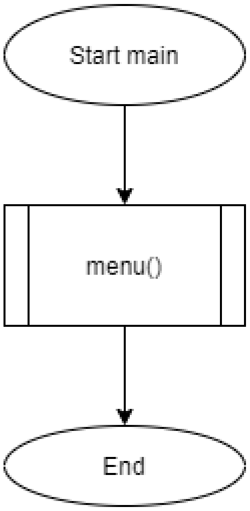


B. Hierarchy Chart

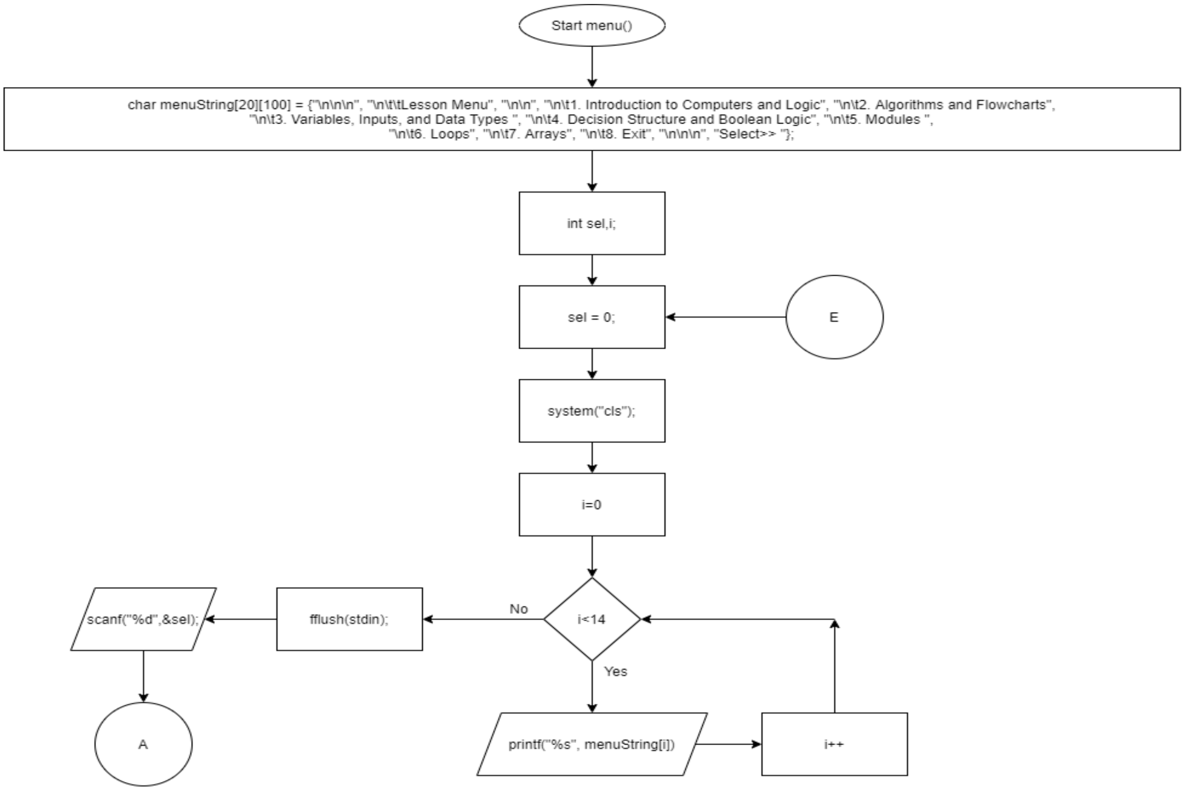


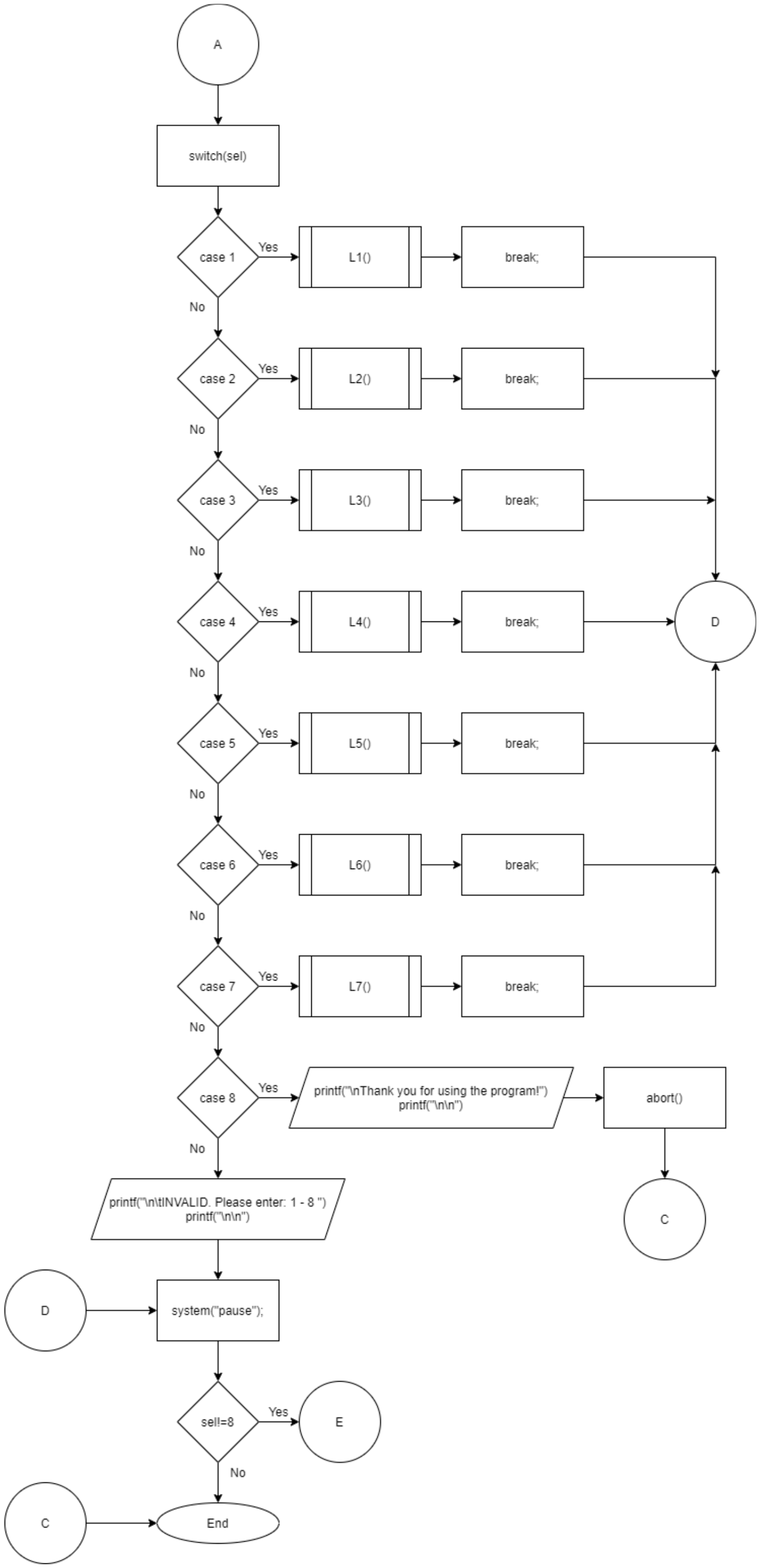
C. Flowchart

As the program output is rather lengthy and repetitive, the flowcharts below only include the main module, the menu module, and the three modules for Lesson 1, Introduction to Computers and Logic. However, the programmers have made flowcharts for all the modules used in the program. (To view them, please refer to this link.)

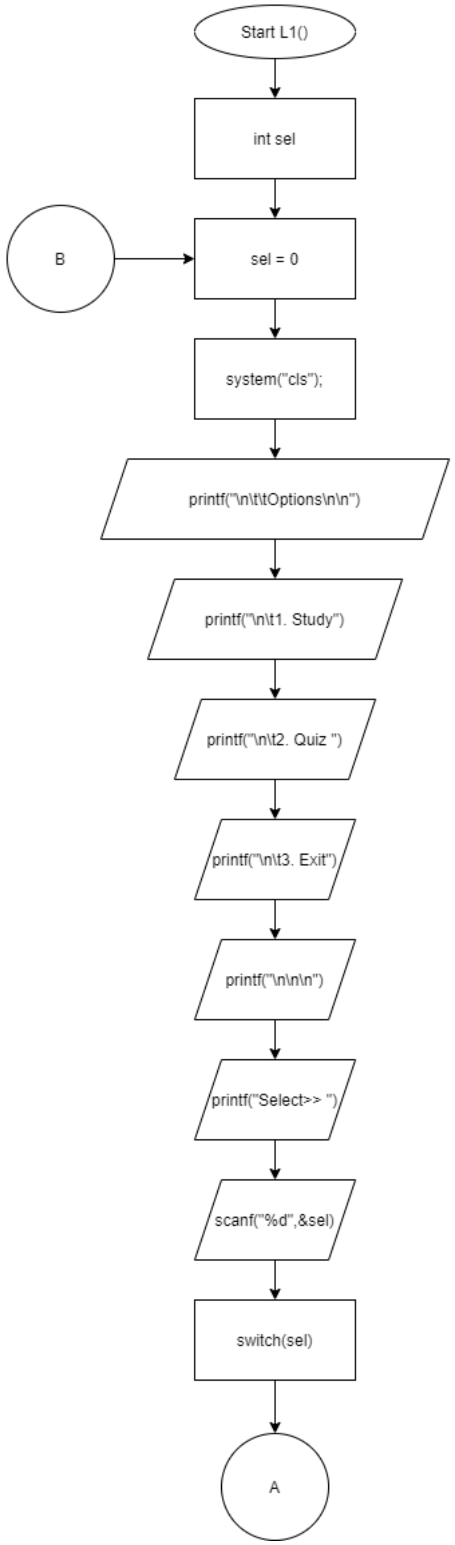
main() Module Flowchart


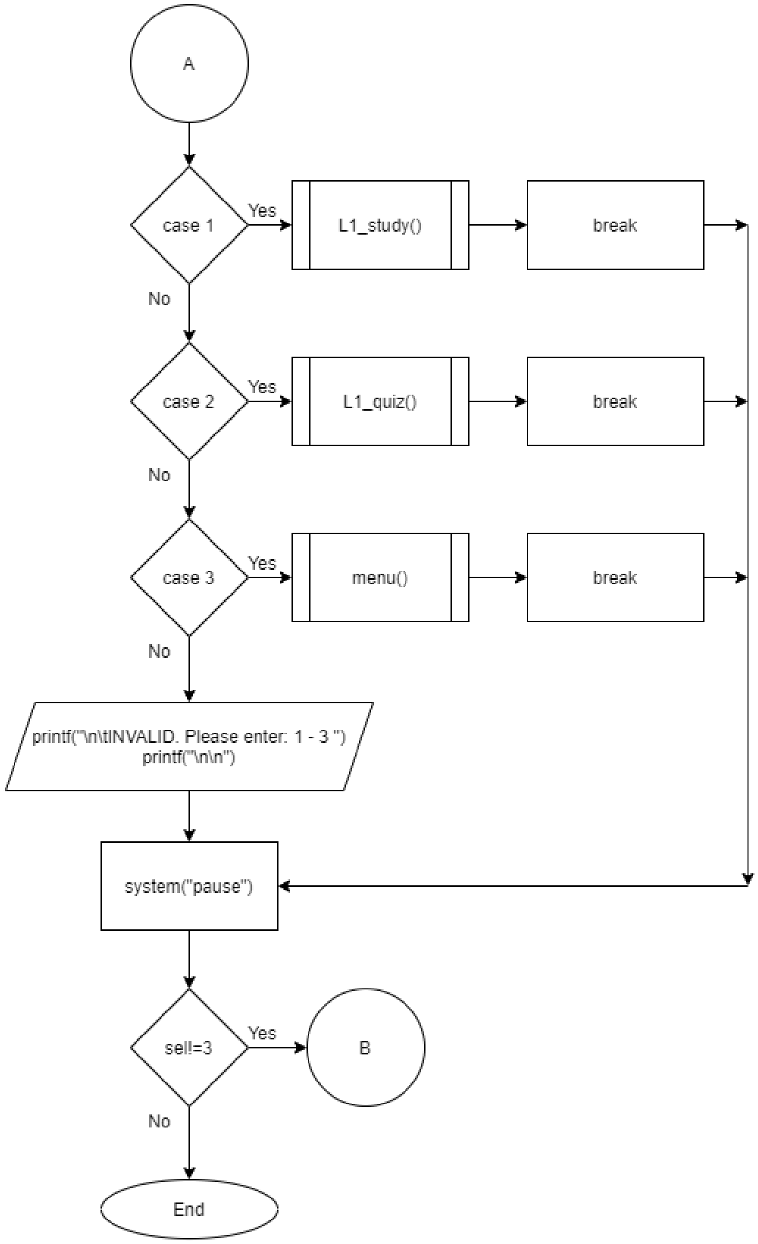
menu() Module Flowchart


Start menu()
char menuString[20][100] = {"\n\n\n", "\n\t\tLesson Menu", "\n\n", "\n\t1. Introduction to Computers and Logic", "\n\t2. Algorithms and Flowcharts",
"\n\t3. Variables, Inputs, and Data Types ", "\n\t4. Decision Structure and Boolean Logic", "\n\t5. Modules ",
"\n\t6. Loops", "\n\t7. Arrays", "\n\t8. Exit", "\n\n\n", "Select>> "};
int sel,i;
sel = 0;
E
system("cls");
i=0
i<14
No
Yes
fflush(stdin);
scanf("%d",&sel);
A
printf("%s", menuString[i])
i++

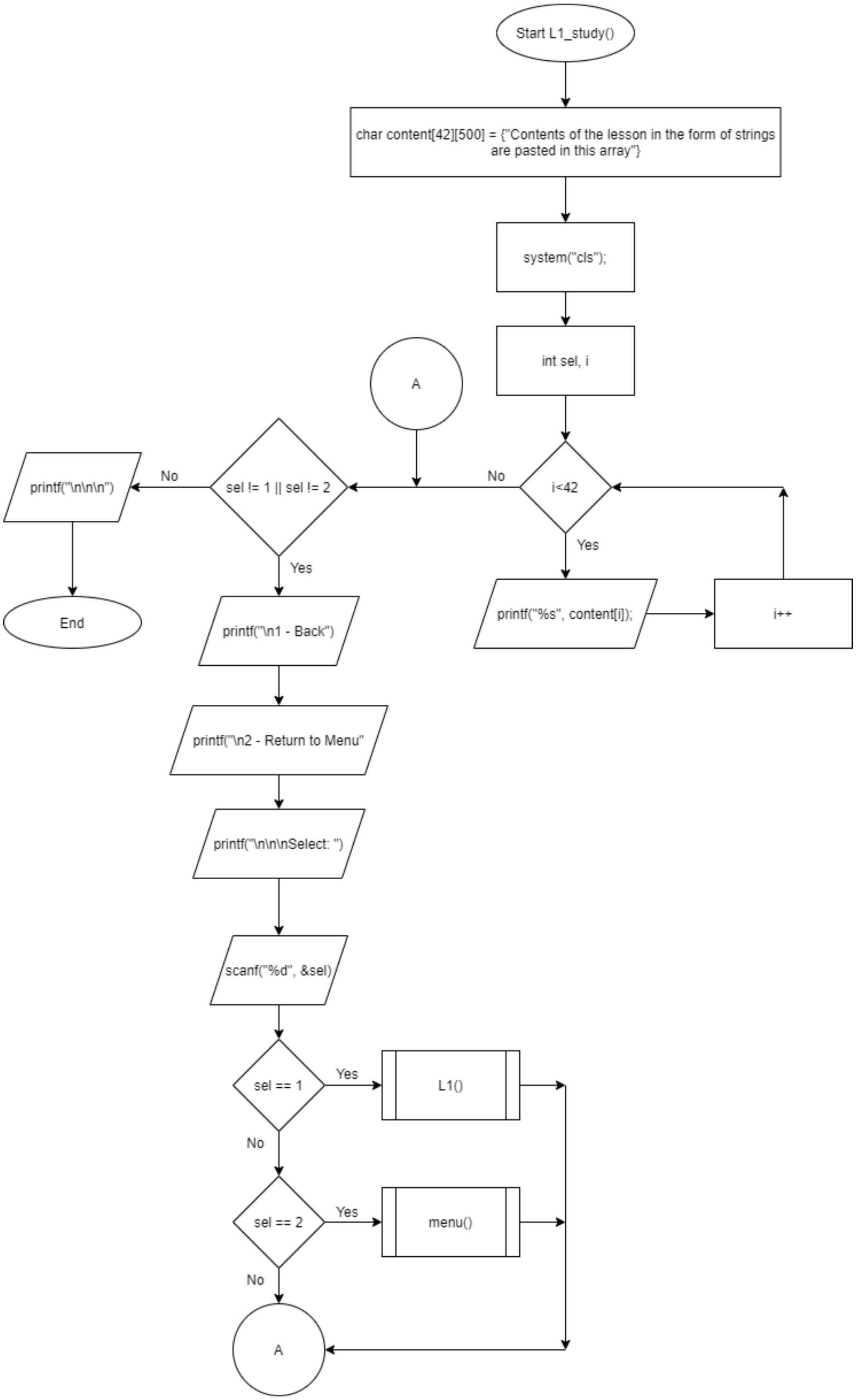

A
switch(sel)
case 1
Yes
L1()
break;
No
case 2
Yes
L2()
break;
No
case 3
Yes
L3()
break;
No
case 4
Yes
L4()
break;
D
No
case 5
Yes
L5()
break;
No
case 6
Yes
L6()
break;
No
case 7
Yes
L7()
break;
No
case 8
Yes
printf("\nThank you for using the program!")
printf("\n\n")
abort()
No
C
printf("\n\tINVALID. Please enter: 1 - 8 ")
printf("\n\n")
D
system("pause");
sel!=8
Yes
E
No
C
End

L1() Module Flowchart

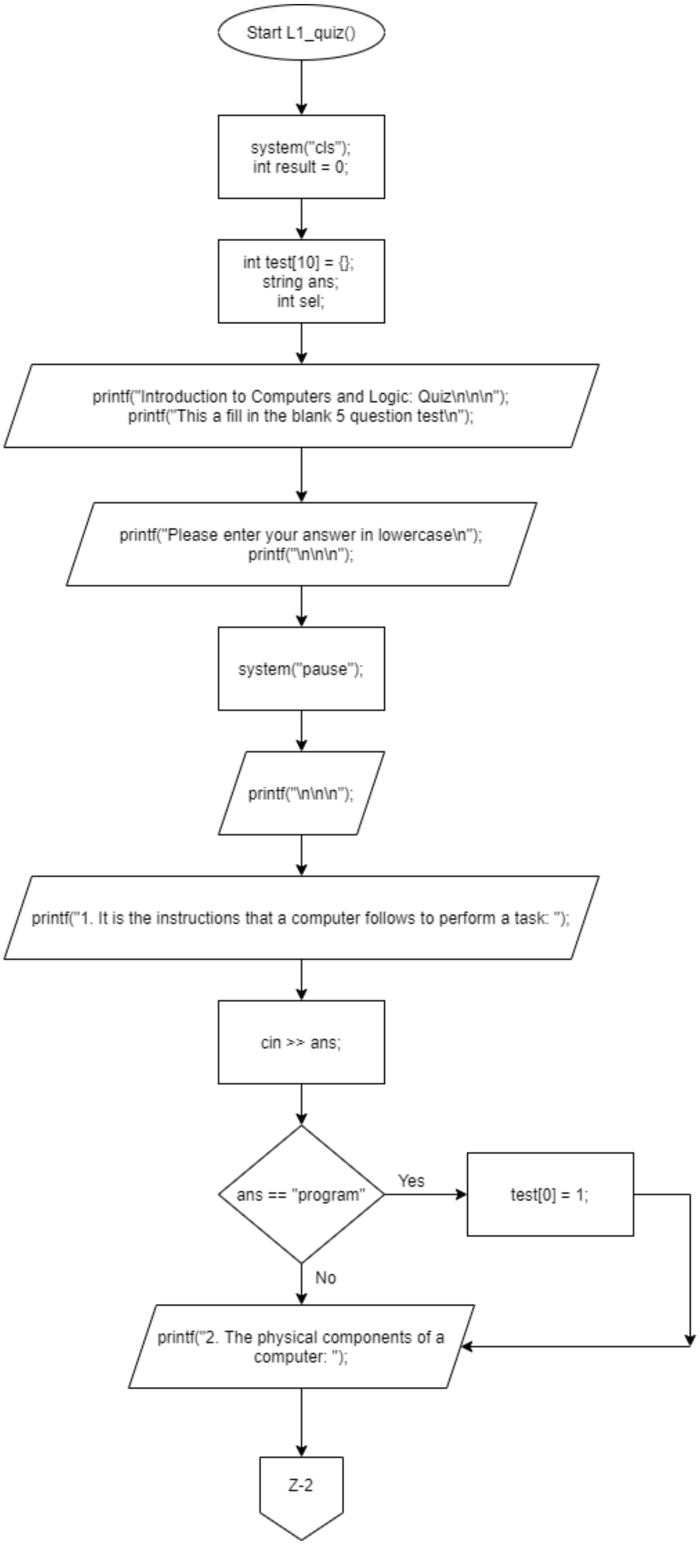


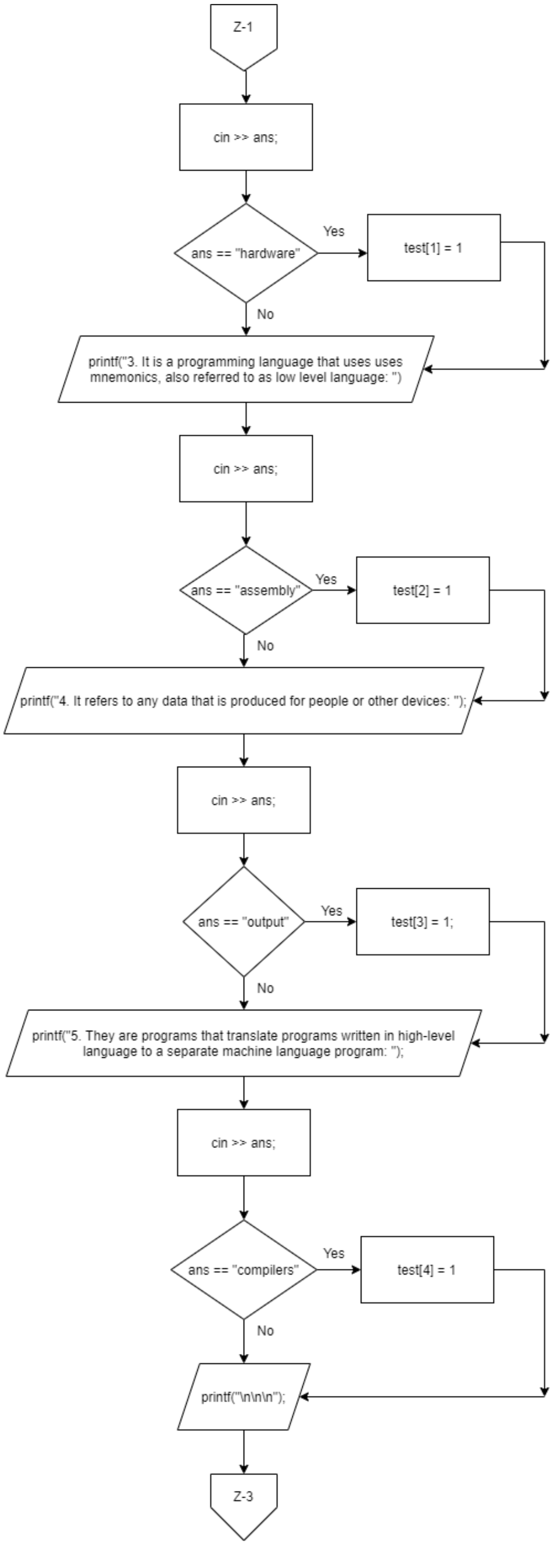

A
case 1
Yes
L1_study()
break
No
case 2
Yes
L1_quiz()
break
No
case 3
Yes
menu()
break
No
printf("\n\tINVALID. Please enter: 1 - 3 ")
printf("\n\n")
system("pause")
sell=3
Yes
B
No
End

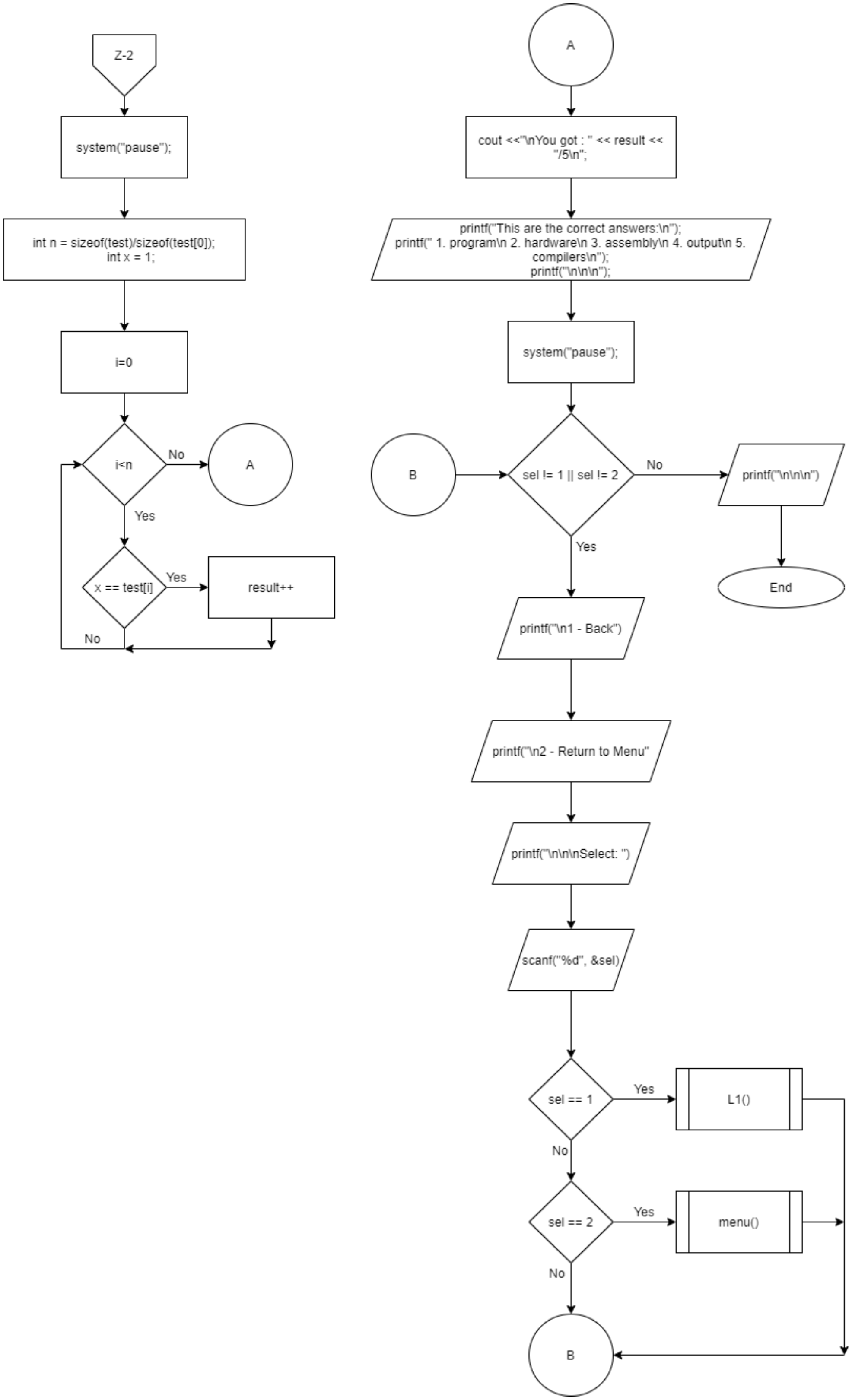
L1_study() Module Flowchart



L1_quiz() Module Flowchart


Start L1_quiz()
system("cls");
int result = 0;
int test[10] = {};
string ans;
int sel;
printf("Introduction to Computers and Logic: Quiz\n\n\n");
printf("This a fill in the blank 5 question test\n");
printf("Please enter your answer in lowercase\n");
printf("\n\n\n");
system("pause");
printf("\n\n\n");
printf("1. It is the instructions that a computer follows to perform a task: ");
cin >> ans;
ans == "program"
Yes
test[0] = 1;
No
printf("2. The physical components of a computer: ");
Z-2


Z-1
cin >> ans;
ans == "hardware"
Yes
test[1] = 1
No
printf("3. It is a programming language that uses uses mnemonics, also referred to as low level language: ")
cin >> ans;
ans == "assembly"
Yes
test[2] = 1
No
printf("4. It refers to any data that is produced for people or other devices: ");
cin >> ans;
ans == "output"
Yes
test[3] = 1;
No
printf("5. They are programs that translate programs written in high-level language to a separate machine language program: ");
cin >> ans;
ans == "compilers"
Yes
test[4] = 1
No
printf("\n\n\n");
Z-3


Z-2
system("pause");
int n = sizeof(test)/sizeof(test[0]);
int x = 1;
i=0
i<n
No
A
Yes
x == test[i]
Yes
result++
No
A
cout <<"\nYou got : " << result << "/5\n";
printf("This are the correct answers:\n");
printf(" 1. program\n 2. hardware\n 3. assembly\n 4. output\n 5. compilers\n");
printf("\n\n\n");
system("pause");
B
sel != 1 || sel != 2
No
printf("\n\n\n")
End
Yes
printf("\n1 - Back")
printf("\n2 - Return to Menu"
printf("\n\n\nSelect: ")
scanf("%d", &sel)
sel == 1
Yes
L1()
No
sel == 2
Yes
menu()
No
B

D. Pseudocode

As the program output is rather lengthy and repetitive, the pseudocode below only includes the essential blocks of code. These include the main module, the menu module, and templates for the three lesson modules. The lesson used in the three lesson templates is Lesson 1, Introduction to Computers and Logic.

```
//the main function of the program where it calls the menu
Module main()
    Call menu()
End Module
```

```
//the function that shows the list of topics in the program
Module menu()
//declaring the string of topics using an array
    Declare Character menuString[20][100]
//declaring variables
    Declare Integer sel, i
//loop to check if the user’s input is still not 1-8
    Do
        Set sel = 0
//displaying the Character menustring for the topics
        For i = 0 to 13
            Display menuString[i]
        End For
//inputing the selected topic based on the number
        Input sel
//case structure for the selection of topic based on the number
        Select sel
            Case 1:
                Call L1()
            Case 2:
                 Call L2()
            Case 3:
                Call L3()
            Case 4:
                Call L4()
            Case 5:
                Call L5()
            Case 6:
                Call L6()
            Case 7:
                Call L7()
            Case 8:
                Display “Thank you for using the program!”
                Exit Program
```

```
                        Default:
                                Display “Invalid. Please enter a number only between 1-8”
                End Select
//end of loop
        While sel != 8
End Module
```

```
//function for option menu Lesson1
Module L1()
        Declare Integer sel
//loop to check if the user’s input is still not 1-3
        Do
        Set sel = 0
                Display "Options"
                Display "1. Study"
                Display "2. Quiz "
                Display "3. Exit"
                Display "Select>> "
//inputing the selected option based on the number
                Input sel
//case structure for the selection of topic based on the number
                Select sel
                        Case 1:
//goes to the lesson 1 study section
                                Call L1_study()
                        Case 2:
//goes to the lesson 1 quiz section
                                Call L1_quiz()
                        Case 3:
//goes back to lesson menu
                                Call menu()
                        Default:
                                Display "INVALID. Please enter: 1 - 3 "
                End Select
//end of loop
        While sel != 3
End Module
```

```
//function for lesson 1 study section
Module L1_study()
//declaring the string of contents using an array
        Declare Character content[42][500]
//declaring variables
        Declare Integer sel, i
//displaying the Character content for the contents of lesson 1
```

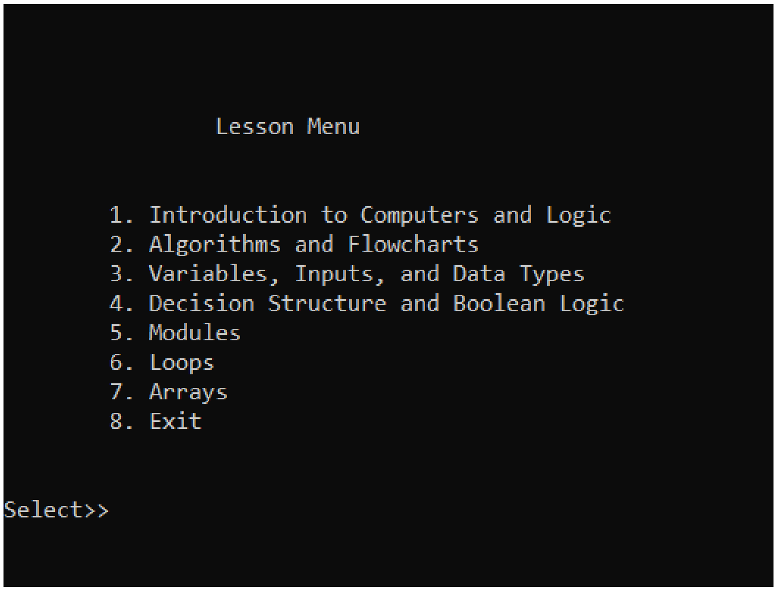
```
    For i = 0 to 42
        Display content[i]
    End For
//loop to check if user’s input is still not equal to 1 or 2
    While sel != 1 OR sel != 2
        Display "1 - Back"
        Display "2 - Return to Menu"
        Display "Select: "
//inputing the selected option based on the number
        Input sel
//goes back to option menu lesson 1
        If sel == 1
            Call L1()
        End If
//goes back to lesson menu
        If sel == 2
            Call menu()
        End If
//end of loop
    End While
End Module
```

```
//function for lesson 1 quiz section
Module L1_quiz()
//declaring variables and an array
    Declare Integer result = 0
    Declare Integer test[5]
    Declare String ans
    Declare Integer sel
    Display "Introduction to Computers and Logic: Quiz"
    Display "This a fill in the blank 5 question test"
    Display "Please enter your answer in lowercase"
//start of quiz
    Display "1. It is the instructions that a computer follows to perform a task: "
//input ans
    Input ans
//check if user’s input is equal to “program”
    If ans == “program”
//if equal, subscript test[0] will be set to 1
//if not, subscript test[0] will be left to 0
        test[0] = 1
    End If
    Display "2. The physical components of a computer: "
//input ans
    Input ans
```

```
//check if user’s input is equal to “hardware”
        If ans == “hardware”
//if equal, subscript test[1] will be set to 1
//if not, subscript test[1] will be left to 0
                test[1] = 1
        Display "3. It is a programming language that uses uses mnemonics, also referred to
        as low level language: "
        Input ans
        If ans == “assembly”
                test[2] = 1
        End If
        Display "4. It refers to any data that is produced for people or other devices: "
        Input ans
        If ans == “output”
                test[3] = 1
        End If
        Display "5. They are programs that translate programs written in high-level language
        to a separate machine language program: "
//input ans
        Input ans
//check if user’s input is equal to “compilers”
        If ans == “compilers”
//if equal, subscript test[4] will be set to 1
//if not, subscript test[4] will be left to 0
                test[4] = 1
        End If
//(size of) is a keyword that gets the size of an array
        int n = sizeof(test)/sizeof(test[0]);
//declaring x = 1, (1) the number that we have to count in array test[]
        int x = 1;
//couting the occurrences of (1) in array test[]
        for (int i=0; i<n; i++)
        if (x == test[i])
//setting the output to int result
        result++;
//displaying the results and the correct answers
        Display “You got :, result,  /5”
        Display "These are the correct answers: "
        Display " 1. program\n 2. hardware\n 3. assembly\n 4. output\n 5. compilers\n”
//loop to check if user’s input is still not equal to 1 or 2
        While sel != 1 OR sel != 2
                Display "1 - Back"
                Display "2 - Return to Menu"
                Display "Select: "
//inputing the selected option based on the number
```

```
                Input sel
//goes back to option menu lesson 1
                If sel == 1
                        Call L1()
                End If
//goes back to lesson menu
                If sel == 2
                        Call menu()
                End If
//end of loop
        End While
End Module
```

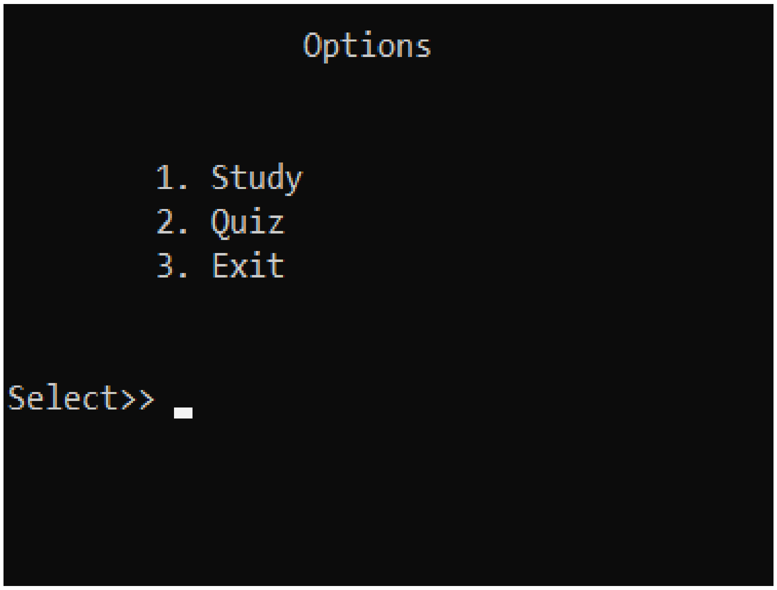
IV. Results



```
                Lesson Menu


        1. Introduction to Computers and Logic
        2. Algorithms and Flowcharts
        3. Variables, Inputs, and Data Types
        4. Decision Structure and Boolean Logic
        5. Modules
        6. Loops
        7. Arrays
        8. Exit


Select>>
```

Figure 4.1. Lesson menu

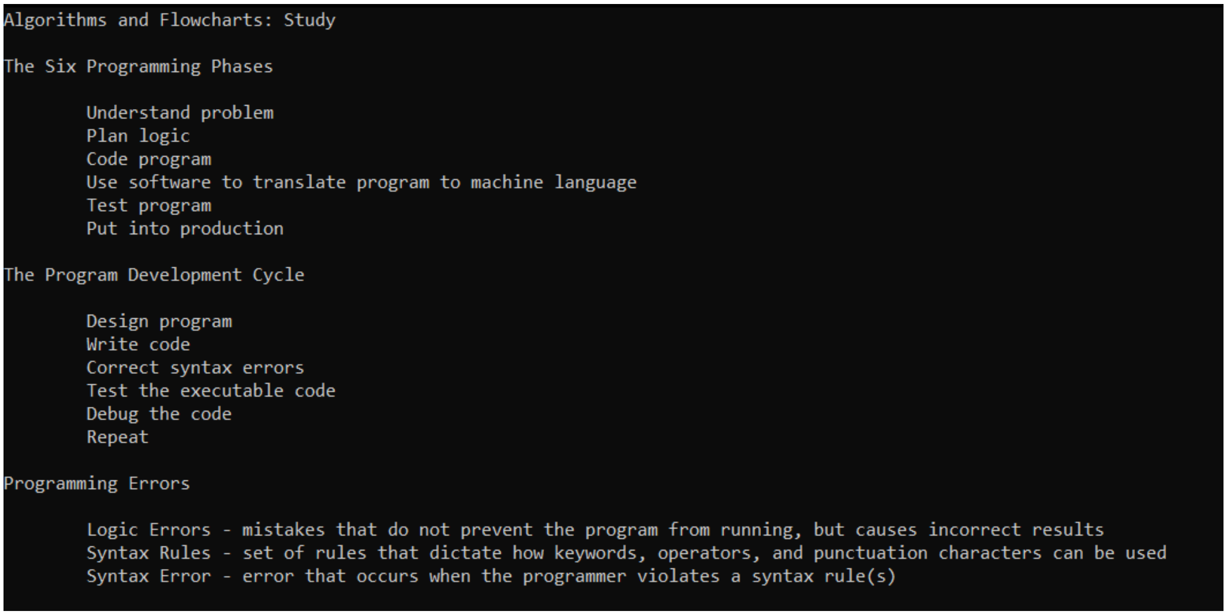
Figure 4.1 is a screenshot of the Lesson Menu, the first screen the user is greeted with when they run the program.



```
                    Options


         1. Study
         2. Quiz
         3. Exit


Select>>
```

Figure 4.2. Option menu per topics

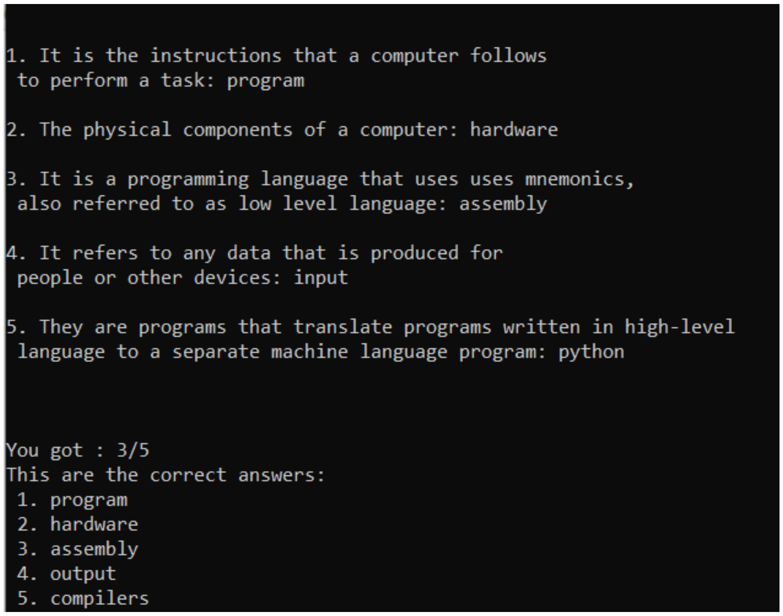
Figure 4.2 is a screenshot of the Options Menu, the screen that the user sees when they select a lesson.



```
Algorithms and Flowcharts: Study

The Six Programming Phases

        Understand problem
        Plan logic
        Code program
        Use software to translate program to machine language
        Test program
        Put into production

The Program Development Cycle

        Design program
        Write code
        Correct syntax errors
        Test the executable code
        Debug the code
        Repeat

Programming Errors

        Logic Errors - mistakes that do not prevent the program from running, but causes incorrect results
        Syntax Rules - set of rules that dictate how keywords, operators, and punctuation characters can be used
        Syntax Error - error that occurs when the programmer violates a syntax rule(s)
```

Figure 5.3. Study section (Topic 2)

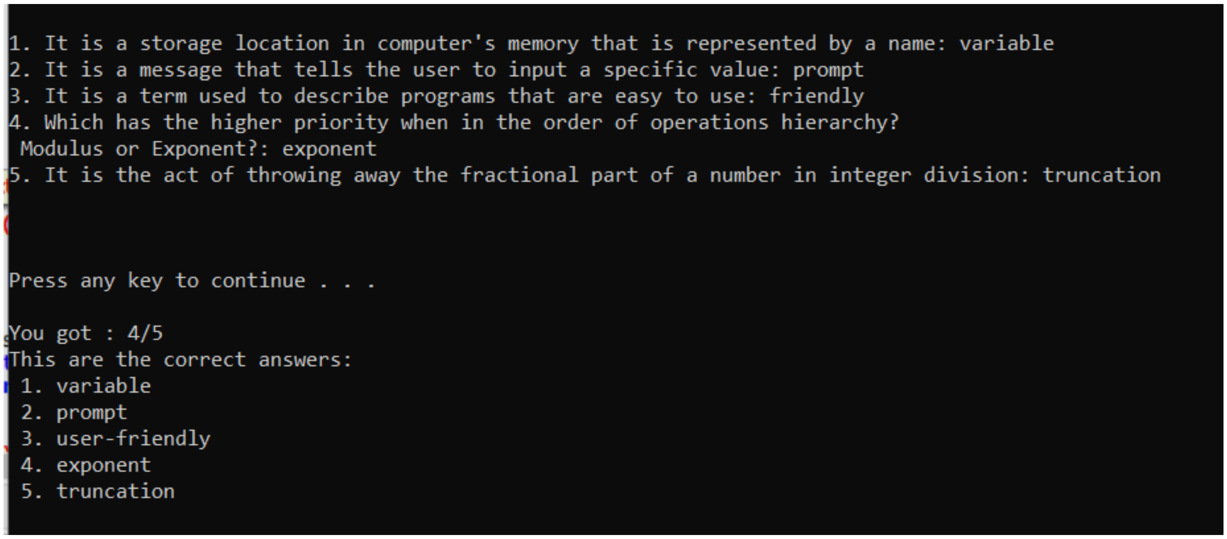
Figure 5.3 is a screenshot of the study screen of the Algorithms and Flowcharts Lesson. The user is brought to this page when they use “1” as their input in Figure 5.2, and if they use “2” as their input in Figure 5.1.



```
1. It is the instructions that a computer follows
 to perform a task: program

2. The physical components of a computer: hardware

3. It is a programming language that uses uses mnemonics,
 also referred to as low level language: assembly

4. It refers to any data that is produced for
 people or other devices: input

5. They are programs that translate programs written in high-level
 language to a separate machine language program: python



You got : 3/5
This are the correct answers:
 1. program
 2. hardware
 3. assembly
 4. output
 5. compilers
```

Figure 5.4. Test run of the quiz program (Topic 1)

Figure 5.4 is a screenshot of the quiz screen of Introduction to Computers and Logic. The user is brought to this page when they use “2” as their input in Figure 5.2, and if they use “1” as their input in Figure 5.1.



```
1. It is a storage location in computer's memory that is represented by a name: variable
2. It is a message that tells the user to input a specific value: prompt
3. It is a term used to describe programs that are easy to use: friendly
4. Which has the higher priority when in the order of operations hierarchy?
 Modulus or Exponent?: exponent
5. It is the act of throwing away the fractional part of a number in integer division: truncation



Press any key to continue . . .

You got : 4/5
This are the correct answers:
 1. variable
 2. prompt
 3. user-friendly
 4. exponent
 5. truncation
```

Figure 5.5. Test run of the quiz program (Topic 3)

Figure 5.5 is a screenshot of the quiz screen of Variables, Inputs, and Data Types. The user is brought to this page when they use “2” as their input in Figure 5.2, and if they use “3” as their input in Figure 5.1.

V. Discussion of Results

The program gives the summary of PROLOGI lessons and is capable of evaluating the quiz accurately, but due to strict limitations, there is no means of testing if the program can help the users' improve their efficiency of the PROLOGI course. The programmers are certain that the program works and includes the specifications that were proposed in the progress report with a few modifications on the content to solve problems in redundancy. However, that is all they can be certain of. There are no means of testing whether the program would be viable outside of testing environments. The researchers lack resources to publish the program and gather data through a series of pre-test and post-test after a select sample used the program. Although this project can serve as a foundation of how specialized programs can be an effective teaching tool especially in the world's new normal.

VI. Analysis, Conclusion, and Future Directives

The program produced from this project is successful in fulfilling the general objective of being helpful for users relearning the fundamental concepts of programming logic and design. Thus, it, in extension, fulfills specific objectives ii and iii as well. However, it is unable to satisfy specific objective i, as its ability to numerically assess the proficiency of the user is questionable, at best. It also has many shortcomings in addition to that. For example, its use is not viable for individuals interested in learning programming logic from scratch. Nonetheless, for students from De La Salle University currently taking up Programming Logic and Design Lecture, it is a comprehensive tool for review purposes as it contains most of the topics discussed in the course.

In the future, as their skills and understanding of programming logic develops, the programmers may consider adding the following features as improvements to the program.

(1) More complex quizzes. At present, the quiz is only limited to 5 items. Furthermore, no matter how many attempts the user makes, the questions will not change, thus every subsequent attempt becomes easier than the last. A more complex quiz would mean not only adding items, but also randomizing their order to discourage the user from memorizing them based on order of appearance rather than associating concept and term.

(2) Moving beyond concepts. The Study section of the program focuses on aiding the user in remembering the concepts discussed in the course. While knowing the definition is important in understanding a concept, applying them for one's self is equally as important. As such, one of the improvements could be the addition of interactive walkthrough tutorials that would gently guide the user as they try to gain a better grasp of programming logic through the association of abstract definitions and how they are practiced.

(3) Expanding the coverage. Programming Logic and Design Lecture is only an introductory course to the world of programming. While this program is the output of the term project for that course, the programmers may continue to develop, update, and add to its contents as they continue to learn about programming through their journey in De La Salle University.

VII. References

Cho, K. W., & Powers, A. (2019). Testing Enhances Both Memorization and Conceptual Learning of Categorical Materials. *Journal of Applied Research in Memory and Cognition*, *8*(2), 166–177. https://doi.org/10.1016/j.jarmac.2019.01.003

Dewhurst, D., Norris, T. & Macleod, H. (2000). Independent student learning aided by computers: An acceptable alternative to lectures? *Computers & Education 35*(3), 223-241. https://doi.org/10.1016/S0360-1315(00)00033-6

Gao, J. & Hargis, J. (2010). Promoting Technology-Assisted Active Learning in Computer Science Education. *The Journal of Effective Teaching, 10,* 81-93.

Harris, J., Bataineh, A., & Bataineh, M. (2016). One to One Technology and its Effect on Student Academic Achievement and Motivation. *Contemporary Educational Technology, 7*(4), 368-381. https://doi.org/10.30935/cedtech/6182

Jaberi, N. (2018). The Use of Computer Programs and Applications by Undergraduates and its Relations to their Motivation toward E-learning and Academic Performance. *International Journal of Education and Literacy Studies, 6*(4), 114-121. https://doi.org/10.7575/aiac.ijels.v.6n.4p.114

Keengwe, J., Onchwari, G., & Wachira, P. (2008). The use of computer tools to support meaningful learning. *AACE Journal, 16(1)*, 77-92.

Kempthorne, D., & Steele, A. (2014). An evaluation of different delivery methods for teaching binary, hex and decimal conversion. *Journal of Applied Computing and Information Technology, 18*(2).

Wagner-Menghin, M., Preusche, I., & Schmidts, M. (2013). The Effects of Reusing Written Test Items: A Study Using the Rasch Model. *ISRN Education*, *2013*, 1–7. https://doi.org/10.1155/2013/585420

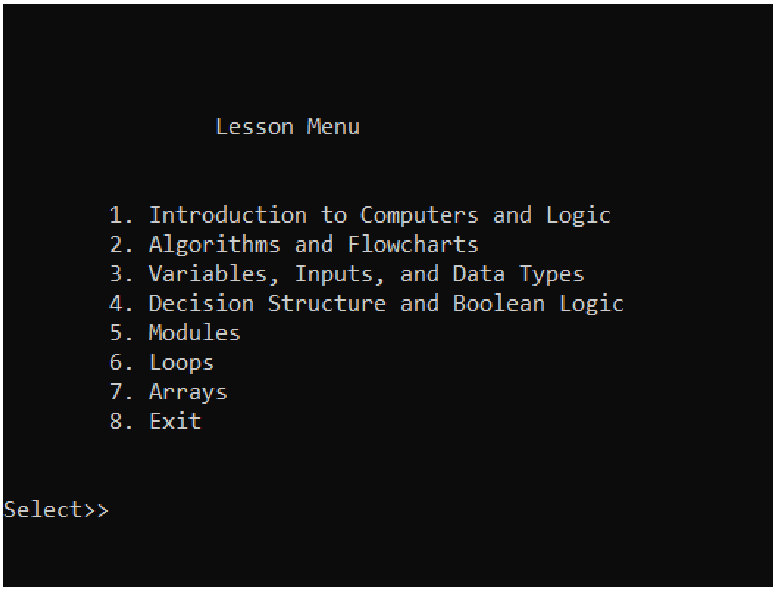
Waraich, A. (2004). Using narrative as a motivating device to teach binary arithmetic and logic gates. *ACM SIGCSE Bulletin, 36*(3), 97-101. https://doi.org/10.1145/1026487.1008024

VIII. Appendices

A. User's Manual

The Pro_Logic: Course Reviewer has a basic interface that is easy to use and convenient for the user. Below are the step by step instructions on how to navigate the program and how to use it properly.

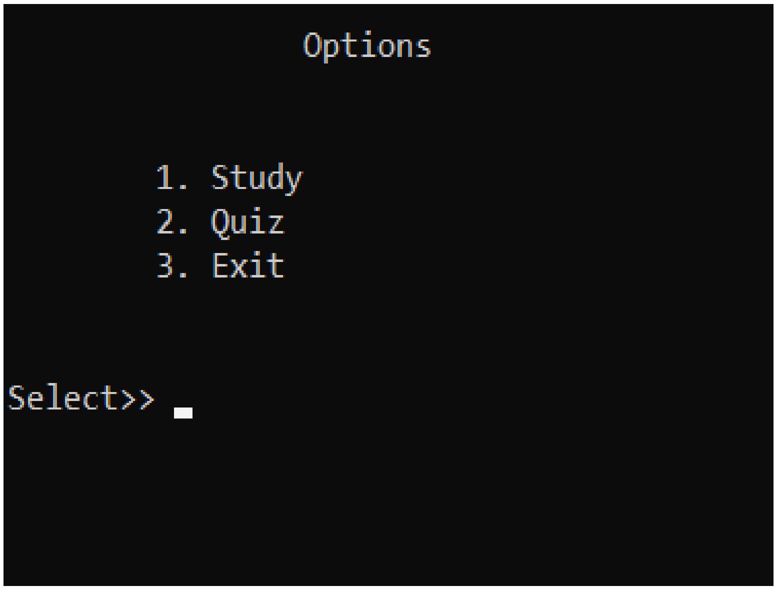
1. The Menu Screen



```
                Lesson Menu


        1. Introduction to Computers and Logic
        2. Algorithms and Flowcharts
        3. Variables, Inputs, and Data Types
        4. Decision Structure and Boolean Logic
        5. Modules
        6. Loops
        7. Arrays
        8. Exit


Select>>
```

After starting, the program will display the menu that shows the available topics. The program will then prompt the user to input the desired lesson. The user can enter the designated number through the select input.

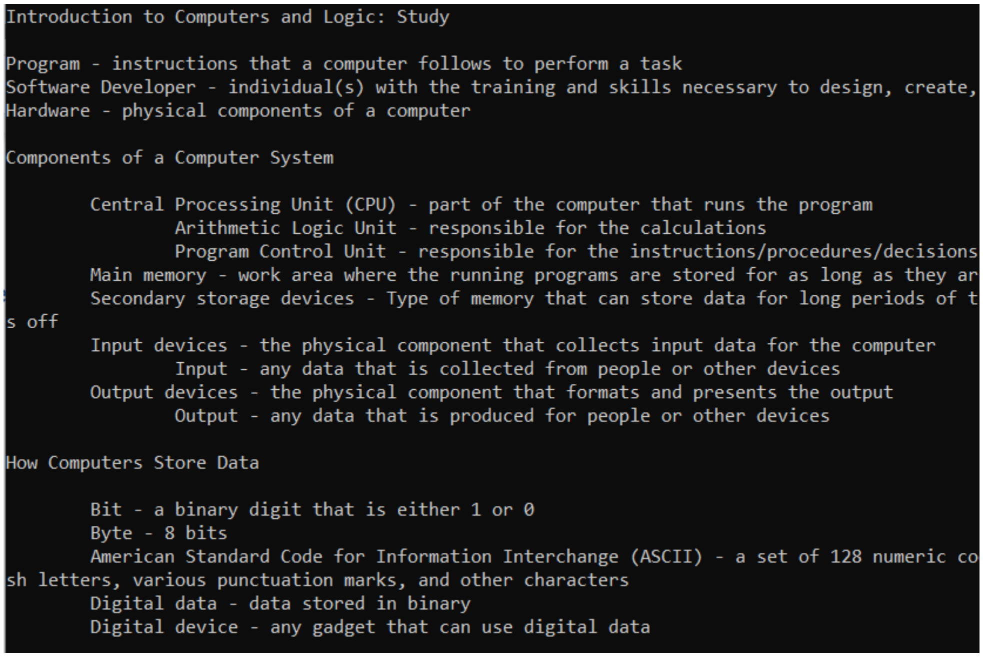
2. The Option Menu



```
                    Options


          1. Study
          2. Quiz
          3. Exit


Select>> _
```

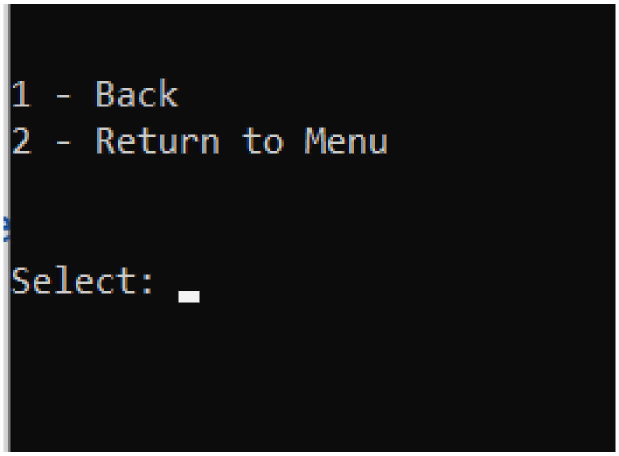
After selecting a topic, the program will then take the user to the option menu. Here the user can choose if he/she wants to go to the study section or the quiz section of that lesson by entering the designated number.

3. Study Section



```
Introduction to Computers and Logic: Study

Program - instructions that a computer follows to perform a task
Software Developer - individual(s) with the training and skills necessary to design, create,
Hardware - physical components of a computer

Components of a Computer System

        Central Processing Unit (CPU) - part of the computer that runs the program
                Arithmetic Logic Unit - responsible for the calculations
                Program Control Unit - responsible for the instructions/procedures/decisions
        Main memory - work area where the running programs are stored for as long as they ar
        Secondary storage devices - Type of memory that can store data for long periods of t
s off
        Input devices - the physical component that collects input data for the computer
                Input - any data that is collected from people or other devices
        Output devices - the physical component that formats and presents the output
                Output - any data that is produced for people or other devices

How Computers Store Data

        Bit - a binary digit that is either 1 or 0
        Byte - 8 bits
        American Standard Code for Information Interchange (ASCII) - a set of 128 numeric co
sh letters, various punctuation marks, and other characters
        Digital data - data stored in binary
        Digital device - any gadget that can use digital data
```

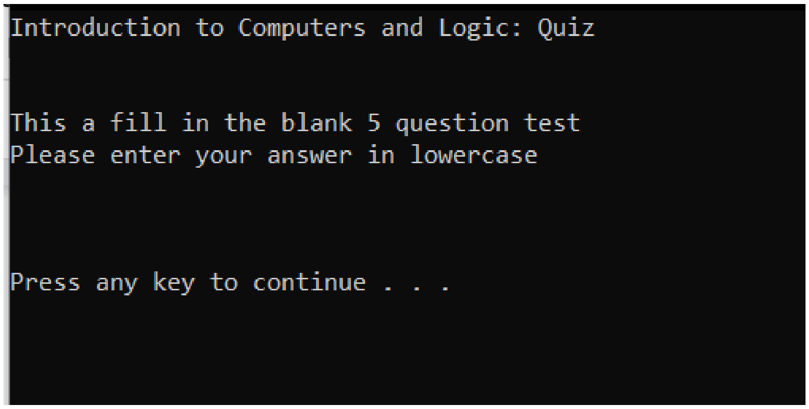
The study section will display the content for the chosen topic. The information here is important because it is the basis of the quiz.



```
1 - Back
2 - Return to Menu


Select: _
```

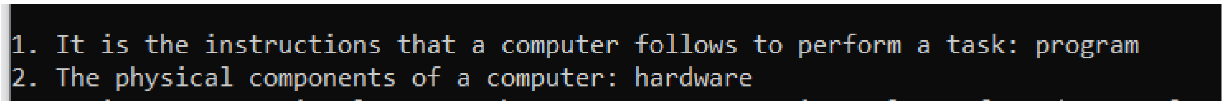
Below the study section, the program will prompt the user if he/she wants to go back to the option or to the lesson menu.

4. Quiz Section



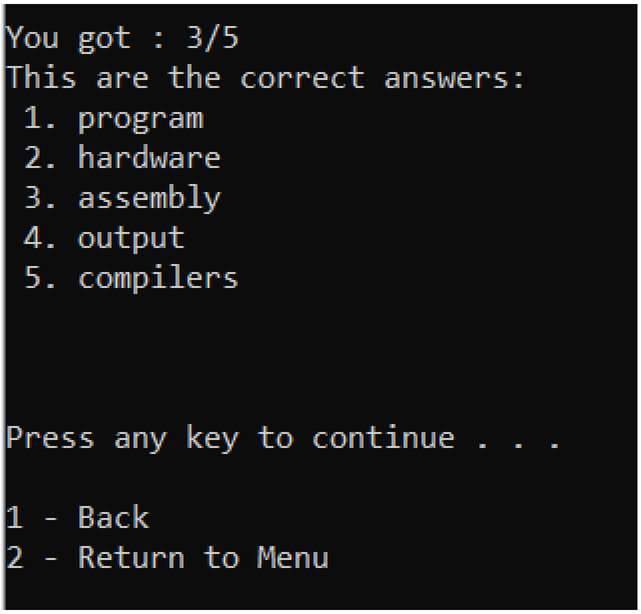
```
Introduction to Computers and Logic: Quiz


This a fill in the blank 5 question test
Please enter your answer in lowercase



Press any key to continue . . .
```

The quiz will be a fill in the blank 5 question test. The input must be in lowercase to properly check the answers.



```
1. It is the instructions that a computer follows to perform a task: program
2. The physical components of a computer: hardware
```

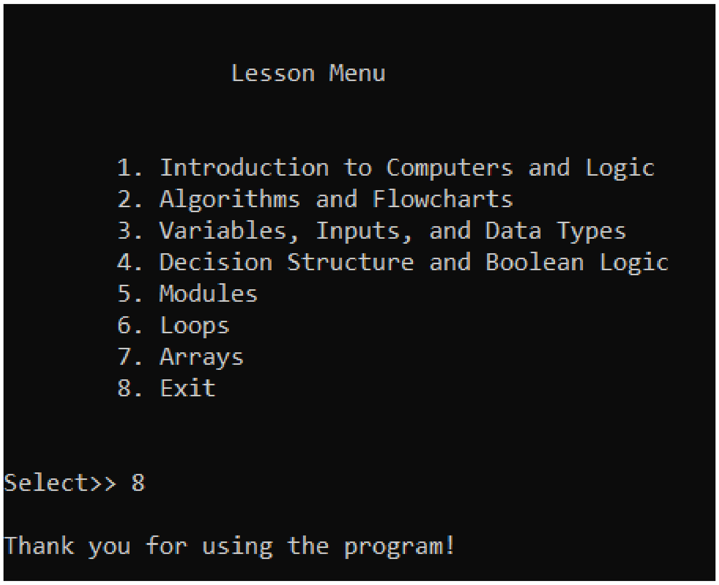
Below is a screenshot of how the program prompts the user for answers.



```
You got : 3/5
This are the correct answers:
 1. program
 2. hardware
 3. assembly
 4. output
 5. compilers



Press any key to continue . . .

1 - Back
2 - Return to Menu
```

After taking the quiz, the program will display the user's score and show the correct answers. The user can then choose to go back to the option or to the lesson menu.

5. Exiting the Program



```
                Lesson Menu


        1. Introduction to Computers and Logic
        2. Algorithms and Flowcharts
        3. Variables, Inputs, and Data Types
        4. Decision Structure and Boolean Logic
        5. Modules
        6. Loops
        7. Arrays
        8. Exit


Select>> 8

Thank you for using the program!
```

To end the program, the user can enter "8" at the lesson menu.

B. Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
```

```
#include <iostream>
using namespace std;
//prototyping functions
void menu();
void L1();
void L1_study();
void L1_quiz();
void L2();
void L2_study();
void L2_quiz();
void L3();
void L3_study();
void L3_quiz();
void L4();
void L4_study();
void L4_quiz();
void L5();
void L5_study();
void L5_quiz();
void L6();
void L6_study();
void L6_quiz();
void L7();
void L7_study();
void L7_quiz();

// calls the menu function that links all other functions
main(){
	menu();
}
// function that asks the user which lesson do they want to take
// through the use of switch case statements
void menu(){
// 2d array that houses the menu strings
	char menuString[20][100] = {"\n\n\n", "\n\t\tLesson Menu", "\n\n", "\n\t1. 
Introduction to Computers and Logic", "\n\t2. Algorithms and Flowcharts", 
								"\n\t3. Variables, Inputs, and 
Data Types ", "\n\t4. Decision Structure and Boolean Logic", "\n\t5. Modules ", 
								"\n\t6. Loops", "\n\t7. Arrays", 
"\n\t8. Exit", "\n\n\n", "Select>> "};
	int sel,i;
// loop that stops when the user chooses int 8; nested loop
	do{
	sel = 0;
	system("cls");
// for loop that exhausts the 2d array menuString; prints everything
        for (i=0;i<14;i++){
                printf("%s", menuString[i]);
        }
        fflush(stdin);
        scanf("%d",&sel);
// conditional statement that determines which function the program should
// call depending on the user
        switch(sel){
                case 1:
                        L1();
                        break;
                case 2:
                        L2();
                        break;
                case 3:
                        L3();
                        break;
                case 4:
                        L4();
                        break;
                case 5:
                        L5();
                        break;
                case 6:
                        L6();
                        break;
                case 7:
                        L7();
                        break;
                case 8:
                        printf("\nThank you for using the program!");
                        printf("\n\n");
                        abort();
                        break;
                default:printf("\n\tINVALID. Please enter: 1 - 8 ");
                                printf("\n\n");
        }
   system("pause");
        }while(sel!=8);
}
//function for lesson 1 option menu
/*The algorithm and logic of L1() is nearly identical to that of L2(), L3(), L4(), L5(), L6(),
and L7(). The only difference being that they will lead to their respective study and quiz
functions. (2)Algorithms and Flowcharts, (3)Variables, Inputs, and Data Types, (4)Decision
```

```
Structure and Boolean Logic, (5)Modules, (6)Loops, and (7)Arrays, respectively.
*/
void L1(){
//declaring variables
        int sel;
//loop to check if user’s input is not between 1-3
        do{
        sel = 0;
        system("cls");
//display the different options
        printf("\n\t\tOptions\n\n");
        printf("\n\t1. Study");
        printf("\n\t2. Quiz ");
        printf("\n\t3. Exit");
        printf("\n\n\n");
        printf("Select>> ");
//get user’s input
        fflush(stdin);
        scanf("%d",&sel);
//case structure for study, quiz, or menu functions
        switch(sel){
                case 1:
//calls lesson 1 study function
                        L1_study();
                        break;
//calls lesson 1 quiz function
                case 2:
                        L1_quiz();
                        break;
//calls menu function
                case 3:
                        menu();
                        break;
                default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                                printf("\n\n");
        }
        system("pause");
//end loop
        }while(sel!=3);
}
/*The algorithm and logic of L1_study() is nearly identical to that of L2_study(),
L3_study(), L4_study(), L5_study(), L6_study, and L7_study(). The only difference being
that they will contain the lessons (2)Algorithms and Flowcharts, (3)Variables, Inputs, and
Data Types, (4)Decision Structure and Boolean Logic, (5)Modules, (6)Loops, and (7)Arrays,
respectively.
```

```
*/
void L1_study(){
//array for the content of lesson 1
        char content[42][500] = {"Introduction to Computers and Logic: Study\n\n",
                                                "Program - instructions that a computer
follows to perform a task\n",
                                                "Software Developer - individual(s)
with the training and skills necessary to design, create, and test computer programs\n",

                                                "Hardware - physical components of a
computer\n\n",
                                                "Components of a Computer
System\n\n",
                                                "\tCentral Processing Unit (CPU) - part
of the computer that runs the program\n",
                                                "\t\tArithmetic Logic Unit - responsible
for the calculations\n",
                                                "\t\tProgram Control Unit - responsible
for the instructions/procedures/decisions\n",

                                                "\tMain memory - work area where the
running programs are stored for as long as they are running\n",
                                                "\tSecondary storage devices - Type of
memory that can store data for long periods of time even when the computer is off\n",
                                                "\tInput devices - the physical
component that collects input data for the computer\n",
                                                "\t\tInput - any data that is collected
from people or other devices\n",
                                                "\tOutput devices - the physical
component that formats and presents the output\n",
                                                "\t\tOutput - any data that is produced
for people or other devices\n\n",
                                                "How Computers Store Data\n\n",
                                                "\tBit - a binary digit that is either 1 or
0\n",

                                                "\tByte - 8 bits\n",
                                                "\tAmerican Standard Code for
Information Interchange (ASCII) - a set of 128 numeric codes that represent the English
letters, various punctuation marks, and other characters\n",
                                                "\tDigital data - data stored in
binary\n",
                                                "\tDigital device - any gadget that can
use digital data\n\n",
                                                "How Programs Work\n\n",
                                        "\tFetch-decode-execute cycle - a
three-step process a computer performs in order to perform a program\n",
                                        "\t\tFetch - the CPU reads the next
instruction from the main memory\n",
                                        "\t\tDecode - the read instruction is
translated so to identify which operation needs to be performed\n",

                                        "\t\tExecute - the decoded instruction
is performed\n\n",
                                        "Programming Laungages\n",
                                        "\tMachine Language - uses binary
numbers\n",
                                        "\tAssembly or Low Level Language -
uses mnemonics\n",
                                        "\tHigh Level Language - uses
keywords, reserved words, or operators\n",
                                        "\tCompilers - programs that translates
programs written in high-level language to a separate machine language program\n",
                                        "\tInterpreters - programs that translate
high-level language programs line by line and promptly executes every instruction right
after it is read\n",
                                        "\tGenerally, compiled programs
execute faster than interpreted programs as the former has already been translated entirely
by the time it was executed\n",

                                        "\tSyntax error - a mistake such as a
misspelled keyword, a missing punctuation character, or the incorrect use of an operator.\n",
                                        "\tIntegrated Development
Environment (IDE) - a text editor that has specialized features for writing statements in a
high-level programming language\n",
                                        "\tTypes of Software\n\n",
                                        "\tSystem Software - programs that
control and manage the basic operations of a computer\n",
                                        "\t\tOperating System (OS) - the most
fundamental set of programs that control the internal operations of the computer hardware,
manages all of the devices connected to the computer,\n",
                                        "\t\tallows data to be saved to and
retrieved from storage devices, and allows other programs to run on the computer\n",
                                        "\t\tUtility Program - performs a
specialized task that enhances the computer’s operation or safeguards data\n",
                                        "\t\tSoftware Development Tools -
programs that programmers use to create, modify, and test software\n",
                                        "\tApplication Software - programs
that make a computer useful for everyday tasks are known as application software, these are
the programs that people normally spend most of their time running on their computers\n\n",
                                                        "\n\n\n"};

        system("cls");
//displays the array of contents
        int sel, i;
        for (i=0;i<42;i++){
                printf("%s", content[i]);
        }
//loop to check if user’s input is not equal to 1 or 2
        while (sel != 1 || sel != 2){
//display options
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
//get user’s input
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
//if user’s input is equal to 1, calls L1() function
                if (sel == 1)
                        L1();
//if user’s input is equal to 2, calls menu() function
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
//function of lesson 1 quiz
/*The algorithm and logic of L1_quiz() is nearly identical to that of L2_quiz(), L3_quiz(),
L4_quiz(), L5_quiz(), L6_quiz(), and L7_study(). The only difference being that they will
contain the questions about (2)Algorithms and Flowcharts, (3)Variables, Inputs, and Data
Types, (4)Decision Structure and Boolean Logic, (5)Modules, (6)Loops, and (7)Arrays,
respectively.
*/

void L1_quiz(){

        system("cls");
        int result = 0;
        int test[10] = {};
        string ans;
        int sel;
//prints instructions for the quiz
        printf("Introduction to Computers and Logic: Quiz\n\n\n");
        printf("This a fill in the blank 5 question test\n");
        printf("Please enter your answer in lowercase\n");
        printf("\n\n\n");
```

```
          system("pause");
          printf("\n\n\n");
// prints the question; scans the user input for the answer; compares answer
// to the correct one; repeat for five items
          printf("1. It is the instructions that a computer follows to perform a task: ");
          cin >> ans;
          if (ans == "program"){
//if user's answer is equal to the correct answer then change test[0] to 1
//repeat to all questions
                    test[0] = 1;
          }
          printf("2. The physical components of a computer: ");
          cin >> ans;
          if (ans == "hardware"){
                    test[1] = 1;
          }
          printf("3. It is a programming language that uses uses mnemonics, also referred to as
low level language: ");
          cin >> ans;
          if (ans == "assembly"){
                    test[2] = 1;
          }
          printf("4. It refers to any data that is produced for people or other devices: ");
          cin >> ans;
          if (ans == "output"){
                    test[3] = 1;
          }
          printf("5. They are programs that translate programs written in high-level language
to a separate machine language program: ");
          cin >> ans;
          if (ans == "compilers"){
                    test[4] = 1;
          }
          printf("\n\n\n");
//end of test
          system("pause");
//check the size of the array
          int n = sizeof(test)/sizeof(test[0]);
//declares x = 1, to count the occurrences of 1 in the test[] array
   int x = 1;
          for (int i=0; i<n; i++)
      if (x == test[i])
//set the output to result
        result++;
//displays the user's score
```

```
   cout <<"\nYou got : " << result << "/5\n";
        printf("This are the correct answers:\n");
//displays the correct answers
        printf(" 1. program\n 2. hardware\n 3. assembly\n 4. output\n 5. compilers\n");
        printf("\n\n\n");
        system("pause");
//loop to check if user’s input is still not equal to 1 or 2
        while (sel != 1 || sel != 2){
//display options
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
//gets user’s input
                scanf("%d", &sel);
//if user’s input is equal to 1, calls L1() function
                if (sel == 1)
                        L1();
//if user’s input is equal to 2, calls menu() function
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L2(){
        int sel;
        do{
        sel = 0;
        system("cls");
        printf("\n\t\tOptions\n\n");
        printf("\n\t1. Study");
        printf("\n\t2. Quiz ");
        printf("\n\t3. Exit");
        printf("\n\n\n");
        printf("Select>> ");
        fflush(stdin);
        scanf("%d",&sel);

        switch(sel){
                case 1:
                        L2_study();
                        break;
                case 2:
                        L2_quiz();
                        break;
                case 3:
```

```
                              menu();
                              break;
                    default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                                        printf("\n\n");
          }
          system("pause");
          }while(sel!=3);
}
void L2_study(){
          char content[23][500] ={
                                                            "Algorithms and Flowcharts: Study\n\n",
                                                            "The Six Programming Phases\n\n",
                                                            "\tUnderstand problem\n\tPlan logic\n\tCode
program\n\tUse software to translate program to machine language\n\tTest program\n\tPut
into production\n\n",
                                                            "The Program Development Cycle\n\n",
                                                            "\tDesign program\n\tWrite code\n\tCorrect
syntax errors\n\tTest the executable code\n\tDebug the code\n\tRepeat\n\n",
                                                            "Programming Errors\n\n",
                                                            "\tLogic Errors - mistakes that do not prevent
the program from running, but causes incorrect results\n",
                                                            "\tSyntax Rules - set of rules that dictate how
keywords, operators, and punctuation characters can be used\n",

                                                            "\tSyntax Error - error that occurs when the
programmer violates a syntax rule(s)\n\n",
                                                            "Algorithm - a set of logical steps that must be
taken to perform a task\n",
                                                            "Flowchart - a visual representation of the
program algorithm that shows the operations used and the sequence in which they are
performed\n\n",
                                                            "Rules in Flowcharting\n\n",
                                                            "\t1) All boxes must be connected by
arrows\n",
                                                            "\t2) There can only be one entry point atop
each symbol, and one exit point at the bottom of all symbols except for the Decision
symbol\n",
                                                            "\t3) The Decision symbol has two exit points
which can be on either side of it or at the bottom and on one side of it\n",
                                                            "\t4) Flowcharts have a top to bottom flow, but
an upward flow is acceptable as long as it does not exceed three symbols\n",

                                                            "\t5) Connectors are used to connect breaks in
the flowchart\n",
                                                            "\t6) Subroutines and Interrupt programs have
```

```
their own and independent flowcharts\n",
                                                "\t7) All flowcharts start with a Terminal or
Predefined Process symbol\n",
                                                "\t8) All flowcharts end with a terminal or
contentious loop\n\n",
                                                "Pseudocode - a program model that follows no
syntax rules and is not meant to be compiled nor executed\n",
                                                "Input-Process-Output - a three-step process
that computer programs typically follow\n\t1) Receive input\n\t2) Process is performed on
the input\n\t3) Produce output\n\n",
                                                "\n\n\n"};
        system("cls");
        int sel,i;

        for (i=0;i<23;i++){
                printf("%s", content[i]);
        }

        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L2();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L2_quiz(){
        system("cls");
        int result = 0;
        int test[10] = {};
        string ans;
        int sel;
        printf("Algorithms and Flowcharts: Quiz\n\n\n");
        printf("This a fill in the blank 5 question test\n");
        printf("Please enter your answer in lowercase\n");
        printf("\n\n\n");
        system("pause");
        printf("\n\n\n");

        printf("1. It is a set of logical steps that must be taken to perform a task: ");
        cin >> ans;
        if (ans == "algorithmm"){
                test[0] = 1;
        }
        printf("2. It is a visual representation of the program algorithm that shows\n the
operations used and the sequence in which they are performed: ");
        cin >> ans;
        if (ans == "flowchart"){
                test[1] = 1;
        }
        printf("3. It is a program model that follows no syntax rules and is not meant to be
compiled nor executed: ");
        cin >> ans;
        if (ans == "pseudocode"){
                test[2] = 1;
        }
        printf("4. It is a three-step process that computer programs typically follow: ");
        cin >> ans;
        if (ans == "input-process-output"){
                test[3] = 1;
        }
        printf("5. It is an error that occurs when the programmer violates a syntax rule(s): ");
        cin >> ans;
        if (ans == "syntax error"){
                test[4] = 1;
        }
        printf("\n\n\n");
        system("pause");
        int n = sizeof(test)/sizeof(test[0]);
   int x = 1;
        for (int i=0; i<n; i++)
      if (x == test[i])
        result++;
   cout <<"\nYou got : " << result << "/5\n";
        printf("This are the correct answers:\n");
        printf(" 1. algorithm\n 2. flowchart\n 3. pseudocode\n 4. input-process-output\n 5.
syntax error\n");
        printf("\n\n\n");
        system("pause");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L2();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L3(){
        int sel;
        do{
        sel = 0;
        system("cls");
        printf("\n\t\tOptions\n\n");
        printf("\n\t1. Study");
        printf("\n\t2. Quiz ");
        printf("\n\t3. Exit");
        printf("\n\n\n");
        printf("Select>> ");
        fflush(stdin);
        scanf("%d",&sel);
        switch(sel){
                case 1:
                        L3_study();
                        break;
                case 2:
                        L3_quiz();
                        break;
                case 3:
                        menu();
                        break;
                default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                        printf("\n\n");
        }
        system("pause");
        }while(sel!=3);
}
void L3_study(){
        char content[46][500] = {"Variables, Inputs, and Data Type: Study\n\n",
                                "Data Types\n",
                                "\tdeclarations for variables that
determine the type and size of the data associated with the variables\n",
                                "\troughly described as numbers,
booleans, characters, arrays, and structures\n\n",
                                "Variable - a storage location in
computer’s memory that is represented by a name\n",
                                "Input - an instruction to read a piece of
data from the keyboard\n\n",
                                    "Rules in Naming Variables\n\n",
                                    "\tOne word names only. No spaces.\n",

                                    "\tMost languages do not allow
punctuation marks in variable names. It is good practice to only make use of alphanumeric
values for naming.\n",
                                    "\tMost languages do not allow the first
character of a variable to be a number\n\n",
                                    "Prompting the User - typically a two
step process that displays a prompt on the screen then reads the value from the
keyboard\n\n",
                                    "\tPrompt - a message that tells the user
to input a specific value\n",
                                    "\tUser-friendly - a term used to
describe programs that are easy to use\n\n",
                                    "Variable Assignments\n\n",
                                    "\tThe value can be the result of a
calculation involving math operators\n",
                                    "\tStatements can be written to store
specific values in variables\n",

                                    "\tValues can be stored in variables
using assignment statements\n",
                                    "\t\tAssignment statement - a construct
that programmers use to set specific values to variables\n",
                                    "\tWhen assigning values to variables,
the variable name will always be on the left side of the = operator. Likewise, the value will
always be on the right side.\n\n",
                                    "Calculations\n\n",
                                    "\tMath expressions perform
calculations and produce a value\n",
                                    "\tThe values on the left and right of
any operator that is not the = operator are called operands\n",
                                    "\tThere are 6 common math operators:
addition(+), subtraction(-), multiplication(*), division(/), modulus(%), and exponent(^)\n\n",
                                    "Order of Operations\n\n",

                                    "\tThe following are the operations
ranked from highest priority(1) to lowest priority(4).\n",
                                    "\t   1) Parentheses\n",
                                    "\t   2) Exponent\n",
                                    "\t   3) Multiplication, Division, and
Modulus\n",
                                    "\t   4) Addition and Subtraction\n\n",
                                    "\tFor 3 and 4, as long as no higher
```

```
priority operation remains, the order in which they are done has no bearing on the output
produced.\n\n",
                                                        "Variable Declaration - a statement that
typically specifies two things about a variable: the name and the data type\n\n",
                                                        "\tMost languages require that variables
be declared before they are used in a program\n",

                                                        "\tUpon declaration, it can be initialized
with a value\n",
                                                        "\t\tVariable Initialization - assigning a
value to a variable\n",
                                                        "\tUninitialized variables are the source
of many errors in programming\n",
                                                        "\t\tUninitialized Variable - a declared
variable that has yet to have been assigned a value\n\n",
                                                        "Numeric Literals - a number that is
written into a program code\n",
                                                        "Truncation - throwing away the
fractional part of a number in integer division\n",
                                                        "Named Constant - a name that
represents a value that cannot be changed during the program execution\n",
                                                        "Hand Tracing (desk checking) - a
simple debugging process for locating hard-to-find errors in a program\n\n",

                                                        "Program Documentation\n\n",
                                                        "\tExternal Documentation - documents
that describe the aspects of a program to the user\n",
                                                        "\tInternal Documentation - comments
that appear in between lines of code written to describe the aspects of a program to the
user\n",
                                                        "\t\tLine comments - comments that
occupy a single line and explain a short section of the program\n",
                                                        "\t\tBlock comments - comments that
occupy several lines and used for lengthy explanations\n",
                                                        "\n\n\n"};
        system("cls");
        int sel,i;

        for (i=0;i<46;i++){
                printf("%s", content[i]);
        }


        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
        printf("\n2 - Return to Menu");
        printf("\n\n\nSelect: ");
        scanf("%d", &sel);
        if (sel == 1)
            L3();
        if (sel == 2)
            menu();
    }
    printf("\n\n\n");
}
void L3_quiz(){
    system("cls");
    int result = 0;
    int test[10] = {};
    string ans;
    int sel;
    printf("Variables, Inputs, and Data Type: Quiz\n\n\n");
    printf("This a fill in the blank 5 question test\n");
    printf("Please enter your answer in lowercase\n");
    printf("\n\n\n");
    system("pause");
    printf("\n\n\n");

    printf("1. It is a storage location in computerâ€™s memory that is represented by a name: ");
    cin >> ans;
    if (ans == "variable"){
        test[0] = 1;
    }
    printf("2. It is a message that tells the user to input a specific value: ");
    cin >> ans;
    if (ans == "prompt"){
        test[1] = 1;
    }
    printf("3. It is a term used to describe programs that are easy to use: ");
    cin >> ans;
    if (ans == "user-friendly"){
        test[2] = 1;
    }
    printf("4. Which has the higher priority when in the order of operations hierarchy? Modulus or Exponent?: ");
    cin >> ans;
    if (ans == "exponent"){
        test[3] = 1;
    }
```

```
        printf("5. It is the act of throwing away the fractional part of a number in integer division: ");
        cin >> ans;
        if (ans == "truncation"){
                test[4] = 1;
        }
        printf("\n\n\n");
        system("pause");
        int n = sizeof(test)/sizeof(test[0]);
   int x = 1;
        for (int i=0; i<n; i++)
      if (x == test[i])
       result++;
   cout <<"\nYou got : " << result << "/5\n";
        printf("This are the correct answers:\n");
        printf(" 1. variable\n 2. prompt\n 3. user-friendly\n 4. exponent\n 5. truncation\n");
        printf("\n\n\n");
        system("pause");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L3();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L4(){
        int sel;
        do{
        sel = 0;
        system("cls");
        printf("\n\t\tOptions\n\n");
        printf("\n\t1. Study");
        printf("\n\t2. Quiz ");
        printf("\n\t3. Exit");
        printf("\n\n\n");
        printf("Select>> ");
        fflush(stdin);
        scanf("%d",&sel);
        switch(sel){
                case 1:
                    L4_study();
                    break;
                case 2:
                    L4_quiz();
                    break;
                case 3:
                    menu();
                    break;
                default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                        printf("\n\n");
        }
        system("pause");
        }while(sel!=3);
}
void L4_study(){
        char content[27][500] = {
                                        "Decision Structure and Boolean Logic:
Study\n\n",
                                        "Control Structure - logical design that controls
the order in which a set of statements executes\n",
                                        "Boolean Expression - a logical statement that
is either true or false\n",
                                        "\tBoolean Variable - a variable that can be
either true or false\n",
                                        "\tBoolean variables are often used as flags\n",
                                        "\t\tFlag - a variable that signals when some
condition exists in the program\n\n",
                                        "Relational Operators - determines whether a
specific relationship exists between two values\n",
                                        "\t1) Greater than (>)\n",

                                        "\t2) Less than (<)\n",
                                        "\t3) Greater than or equal to (>=)\n",
                                        "\t4) Less than or equal to (<=)\n",
                                        "\t5) Equal to (==)\n",
                                        "\tNot equal to (!=)\n\n",
                                        "Logical Operators can be used to create
complex boolean expressions or compound expressions\n",
                                        "\t1) AND - both subexpressions must be true
for the expression to be true\n",
                                        "\t2) OR - at least one of the subexpressions
must be true for the expression to be true\n",

                                        "\t3) NOT - both subexpressions must be false
for the expression to be true\n\n",
                                                  "Decision Structure (Selection Structure) -
allows a program to perform actions only under certain conditions\n",
                                                  "\tSingle alternative - a decision structure with
only one possible path of execution as the other path exits the structure\n",
                                                  "\tDual alternative - a decision structure that
has two possible paths of execution - one path is taken if a condition is true, and the other is
taken if the condition is false\n",
                                                  "\tNested - a decision structure inside another
decision structure that would allow for the testing of more than one condition\n",
                                                  "\tCase Structure - a decision structure which
uses the value or a variable or an expression to determine the path of execution\n",
                                                  "\t\tCase structures are multiple alternative
decision structures\n",
                                                  "\t\tCase structures follow a top to bottom
flow\n",

                                                  "\t\tTest Expression - a single variable or
expression that is compared to each case statement\n",
                                                  "\t\t\tIf no match is found between the test
expression and the case statement, the default statements are used instead\n",
                                                  "\n\n\n"};
        system("cls");
        int sel, i;
        for (i=0;i<27;i++){
                printf("%s", content[i]);
        }

        printf("\n\n\n");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L4();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L4_quiz(){
        system("cls");
        int result = 0;
        int test[10] = {};
        string ans;
        int sel;
        printf("Decision Structure and Boolean Logic: Quiz\n\n\n");
        printf("This a fill in the blank 5 question test\n");
        printf("Please enter your answer in lowercase\n");
        printf("\n\n\n");
        system("pause");
        printf("\n\n\n");

        printf("1. It is a variable that signals when some condition exists in the program: ");
        cin >> ans;
        if (ans == "flag"){
                test[0] = 1;
        }
        printf("2. It is a logical operator that requires both subexpressions to be true for the
expression to be true: ");
        cin >> ans;
        if (ans == "and"){
                test[1] = 1;
        }
        printf("3. It is a logical operator that requires at least one of the subexpressions to be
true for the expression to be true: ");
        cin >> ans;
        if (ans == "or"){
                test[2] = 1;
        }
        printf("4. It is a logical operator that requires both subexpressions to be false for the
expression to be true: ");
        cin >> ans;
        if (ans == "not"){
                test[3] = 1;
        }
        printf("5. It is a decision structure inside another decision structure that would allow
for the testing of more than one condition: ");
        cin >> ans;
        if (ans == "nested"){
                test[4] = 1;
        }
        printf("\n\n\n");
        system("pause");
        int n = sizeof(test)/sizeof(test[0]);
   int x = 1;
        for (int i=0; i<n; i++)
      if (x == test[i])
        result++;
   cout <<"\nYou got : " << result << "/5\n";
```

```
        printf("This are the correct answers:\n");
        printf(" 1. flag\n 2. and\n 3. or\n 4. not\n 5. nested\n");
        printf("\n\n\n");
        system("pause");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L4();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L5(){
        int sel;
        do{
        sel = 0;
        system("cls");
        printf("\n\t\tOptions\n\n");
        printf("\n\t1. Study");
        printf("\n\t2. Quiz ");
        printf("\n\t3. Exit");
        printf("\n\n\n");
        printf("Select>> ");
        fflush(stdin);
        scanf("%d",&sel);
        switch(sel){
                case 1:
                        L5_study();
                        break;
                case 2:
                        L5_quiz();
                        break;
                case 3:
                        menu();
                        break;
                default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                                printf("\n\n");
        }
        system("pause");
        }while(sel!=3);
}
```

```
void L5_study(){
        char content[28][500] = {
                                        "Modules: Study\n\n",
                                        "Module - a group of statements that exist
within a program for the purpose of performing a specific task\n",
                                        "\tThe use of modules is an approach often
referred to as divide and conquer as a large task in divided into smaller tasks that are easier
to accomplish\n\tOther names for modules are procedures, subroutines, subprograms,
methods, and functions\n\n",
                                        "Benefits of Using Modules\n",
                                        "\tSimpler Code - several small modules are
easier to read than one long sequence of statements\n",
                                        "\tCode Reuse - writing a module for a
recurring block of code and executing said module whenever it is needed again makes for
not only a cleaner program but also convenience for the programmer\n",
                                        "\tBetter Testing - programmers can test each
module individually to locate the error and fix it\n",
                                        "\tModular programming - a programming
technique that involves separating the program into independent and interchangeable
modules each capable of an aspect of the desired functionality\n",

                                        "\tFaster Development and Easier Facilitation
of Teamwork - The application of divide and conquer where individual programmers are
assigned to work on different modules to avoid duplication of outputs and maximize
productivity\n\n",
                                        "Defining a Module\n",
                                        "\tModule Definition - a code for a
module\n\t\tHeader - starting point of the module\n\t\tBody - list of statements that belong to
the module\n",
                                        "\tModule definitions specify what modules do,
but they do not cause the modules to execute\n\n",
                                        "Calling a Module\n",
                                        "\tTo execute modules, they must be called\n",
                                        "\tWhen a module is called, the computer
jumps to that module and executes the statements in the module's body\n",
                                        "\tThen, when the end of the module is reached,
the computer jumps back to the part of the program that called the module, and the program
resumes execution at that point\n\n",

                                        "Hierarchy Chart (Structure Chart) - boxes that
represent each module in the program\n\tLocal Variables - a variable declared within a
module and only exists within said module\n",
                                        "\tDifferent modules may have local variables
that have the same name\n",
                                        "\tErrors will occur if a statement in one
module tries to access a local variable that belongs to another module\n",
                                        "\tScope - a term used to describe the part of
the program in which a variable may be accessed\n",
                                        "\t\tA variable is visible only to statements
inside the variable’s scope\n",
                                        "\t\tA local variable’s scope usually begins at
the variable’s declaration and ends at the end of the module in which the variable is
declared\n",
                                        "\t\tThe variable cannot be accessed by
statements that are outside its scope\n\n",
                                        "Argument - any piece of data that is passed
into a module when a module is called\n",

                                        "Parameter Variable (parameter) - a special
variable that receives an argument when a module is called\n",
                                        "\tSome languages allow you to pass an
argument into a parameter variable of a different type as long as no data will be lost\n",
                                        "Return Point - the memory address of the
location in the program the computer should return to after visiting a module\n",
                                        "\n\n\n"};
        system("cls");
        int sel, i;

        for (i=0;i<28;i++){
                printf("%s", content[i]);
        }

        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L5();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L5_quiz(){
        system("cls");
        int result = 0;
        int test[10] = {};
        string ans;
        int sel;
```

```
          printf("Modules: Quiz\n\n\n");
          printf("This a fill in the blank 5 question test\n");
          printf("Please enter your answer in lowercase\n");
          printf("\n\n\n");
          system("pause");
          printf("\n\n\n");

          printf("1. It is a group of statements that exist within a program for the purpose of
performing a specific task: ");
          cin >> ans;
          if (ans == "module"){
                    test[0] = 1;
          }
          printf("2. It is the starting point of a module: ");
          cin >> ans;
          if (ans == "header"){
                    test[1] = 1;
          }
          printf("3. It is a list of statements that belong to a module: ");
          cin >> ans;
          if (ans == "body"){
                    test[2] = 1;
          }
          printf("4. It is the action used to let the program know to execute a module: ");
          cin >> ans;
          if (ans == "call"){
                    test[3] = 1;
          }
          printf("5. Any piece of data that is passed into a module when a module is called: ");
          cin >> ans;
          if (ans == "argument"){
                    test[4] = 1;
          }
          printf("\n\n\n");
          system("pause");
          int n = sizeof(test)/sizeof(test[0]);
   int x = 1;
          for (int i=0; i<n; i++)
      if (x == test[i])
        result++;
   cout <<"\nYou got : " << result << "/5\n";
          printf("This are the correct answers:\n");
          printf(" 1. program\n 2. header\n 3. body\n 4. call\n 5. argument\n");
          printf("\n\n\n");
          system("pause");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L5();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L6(){
        int sel;
        do{
        sel = 0;
        system("cls");
        printf("\n\t\tOptions\n\n");
        printf("\n\t1. Study");
        printf("\n\t2. Quiz ");
        printf("\n\t3. Exit");
        printf("\n\n\n");
        printf("Select>> ");
        fflush(stdin);
        scanf("%d",&sel);
        switch(sel){
                case 1:
                        L6_study();
                        break;
                case 2:
                        L6_quiz();
                        break;
                case 3:
                        menu();
                        break;
                default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                                printf("\n\n");
        }
        system("pause");
        }while(sel!=3);
}
void L6_study(){
        char content[22][500] = {
                                        "Loops: Study\n\n",
                                        "Repetition Structures - a structure that causes a
statement or set of statements to execute repeatedly\n\n",
                                        "Condition-Controlled Loop - uses a true or false
condition to control the decision for the execution of another iteration\n",
                                        "\tWhile - a repetition structure that repeats as long as
the condition is true\n\t\t1) A condition is tested  for a boolean value\n\t\t2) The statement(s)
repeats for as long as the condition is true\n",
                                        "\tDo-While - a repetition structure that repeats as long
as the condition is true\n\t\t1) The statement(s) is executed\n\t\t2) The condition is tested for
a boolean value and for as long as the result is true, the structure repeats\n",
                                        "\tDo-Until - a repetition structure that repeats until a
condition is true\n\t\t1) The statement(s) is executed\n\t\t2) The condition is tested for a
boolean value and when the result is true, the structure ends\n\n",
                                        "Count-Controlled Loop - uses a counter to control the
number of iterations\n",
                                        "\tCounter Variable - name of a variable that is used as
a counter\n",

                                        "\tStarting Value - the initial value of the counter\n",
                                        "\tMax Value - the maximum value of the counter\n",
                                        "\tProcess of a Count-Controlled Loop\n",
                                        "\t\t1) Initialization - the counter variable is given a
starting value\n",
                                        "\t\t2) Test - the counter variable is compared to the
maximum value, when it is less than or equal, the loop iterates, and if not, the program exits
the loop\n",
                                        "\t\t3) Increment - the value of the counter variable is
increased by one\n",
                                        "\tStep Amount - the amount by which the counter
variable is incremented\n",
                                        "\tDecrement - decreasing the value of the counter
variable by one\n",

                                        "\tAn example of a count-controlled loop is the For
Statement\n\n",
                                        "Sentinels - special values that mark the end of a list of
values\n",
                                        "Nested Loops - a loops within other loops\n",
                                        "Accumulator - a variable used to keep the running
total of a loop\n",
                                        "Running Total - a sum of numbers that accumulates
with each iteration of a loop\n",
                                        "\n\n\n"};
        system("cls");
        int sel,i;
        for (i=0;i<22;i++){
                printf("%s", content[i]);
        }

        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L6();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L6_quiz(){
        system("cls");
        int result = 0;
        int test[10] = {};
        string ans;
        int sel;
        printf("Loops: Quiz\n\n\n");
        printf("This a fill in the blank 5 question test\n");
        printf("Please enter your answer in lowercase\n");
        printf("\n\n\n");
        system("pause");
        printf("\n\n\n");

        printf("1. It is a repetition structure that tests the condition for a boolean value before
performing\n ");
        printf("the instruction(s) under it and proceeds to repeat the process for as long as the
condition is true: ");
        cin >> ans;
        if (ans == "while"){
                test[0] = 1;
        }
        printf("2. It is a repetition structure that performs the instruction(s) then tests the
condition for a\n ");
        printf("boolean value and proceeds to repeat the process for as long as the condition
is true: ");
        cin >> ans;
        if (ans == "do-while"){
                test[1] = 1;
        }
        printf("3. It is a repetition structure that uses a counter to control the number of
iterations: ");
        cin >> ans;
        if (ans == "count-controlled"){
                test[2] = 1;
        }
        printf("4. It is the increasing of the value of the counter variable by one: ");
        cin >> ans;
        if (ans == "increment"){
                test[3] = 1;
        }
        printf("5. It is a special value that mark the end of a list of values: ");
        cin >> ans;
        if (ans == "sentinel"){
                test[4] = 1;
        }
        printf("\n\n\n");
        system("pause");
        int n = sizeof(test)/sizeof(test[0]);
   int x = 1;
        for (int i=0; i<n; i++)
     if (x == test[i])
       result++;
   cout <<"\nYou got : " << result << "/5\n";
        printf("This are the correct answers:\n");
        printf(" 1. while\n 2. do-while\n 3. count-controlled\n 4. increment\n 5. sentinel\n");
        printf("\n\n\n");
        system("pause");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Lesson Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L6();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
void L7(){
        int sel;
        do{
        sel = 0;
        system("cls");
        printf("\n\t\tOptions\n\n");
```

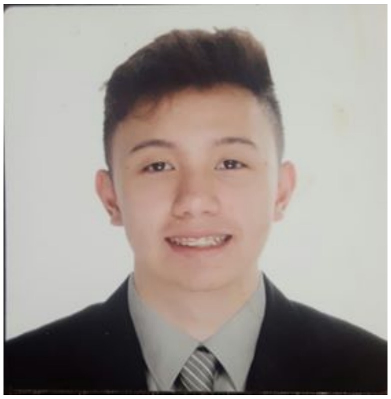
```
    printf("\n\t1. Study");
    printf("\n\t2. Quiz ");
    printf("\n\t3. Exit");
    printf("\n\n\n");
    printf("Select>> ");
    fflush(stdin);
    scanf("%d",&sel);
    switch(sel){
        case 1:
            L7_study();
            break;
        case 2:
            L7_quiz();
            break;
        case 3:
            menu();
            break;
        default:printf("\n\tINVALID. Please enter: 1 - 3 ");
                printf("\n\n");
    }
    system("pause");
    }while(sel!=3);
}
void L7_study(){
    char content[36][500] = {
                    "Arrays: Study\n\n",
                    "Array - a data structure consisting of a collection of
elements, each identifiable through its assigned subscript\n",
                    "\tElement - storage locations in an array\n",
                    "\tSubscript - a unique number assigned to each
element in an array\n",
                    "\tSize declarator - a number inside the brackets,
specifies the number of values that the array can hold\n\n",
                    "Array Initialization -  assigning values to the elements
of an array\n",
                    "Array Bounds Checking - a process that ensures no
invalid subscripts are used in a program\n",
                    "Off-by-One Errors - Error that occurs when a loop
does one iteration too many or too little\n\n",

                    "Partially Filled Arrays\n",
                    "\tNormally used with an accompanying integer
variable that holds the number of items that are actually stored in the array\n",
                    "\tAfter an element is initialized, the variable is
incremented\n",
                                        "\tWhen the program steps through the array’s
elements, the value of this variable is used instead of the array’s size to determine the
maximum subscript\n\n",
                                        "For Each Loop - a specialized version of the For loop
that can simplify array processing by stepping through every element in an array and
retrieving its value\n\n",
                                        "Sequential Search Algorithm - a technique for finding
an item in an array\n",
                                        "\t1) It uses a loop to sequentially step through an
array, starting with the first element.\n",
                                        "\t2) It compares each element with the value being
searched for and stops when the value is found or the end of the array is encountered\n",

                                        "\t3) If the value being searched for is not in the array,
the algorithm unsuccessfully searches to the end of the array\n\n",
                                        "Parallel Arrays -  two or more arrays that hold related
data, and the related elements in each array are accessed with a common subscript\n\n",
                                        "One-Dimensional Arrays - capable of holding only
one set of data\n",
                                        "Two-Dimensional Arrays - capable of holding
multiple sets of data on account of having rows and columns\n",
                                        "Three-Dimensional Arrays - capable of holding
multiple sets of data on account of having another dimension aside from the rows and
columns of a two-dimensional array\n\n",


                                        "Copying Arrays\n",
                                        "\tThe value of each variable must be copied
separately\n",
                                        "\tUsually done with a loop and parallel arrays\n\n",
                                        "Sorting Algorithm - rearranges the contents of an
array so they appear in a specific order (i.e. bubble sort)\n",
                                        "\n\n\n"};
        system("cls");
        int sel, i;

        for (i=0;i<36;i++){
                printf("%s", content[i]);
        }

        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Lesson Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                L7();
            if (sel == 2)
                menu();
        }
        printf("\n\n\n");
}
void L7_quiz(){
        system("cls");
        int result = 0;
        int test[10] = {};
        string ans;
        int sel;
        printf("Arrays: Quiz\n\n\n");
        printf("This a fill in the blank 5 question test\n");
        printf("Please enter your answer in lowercase\n");
        printf("\n\n\n");
        system("pause");
        printf("\n\n\n");

        printf("1. It is a data structure consisting of a collection of elements,\n each
identifiable through its assigned subscript: ");
        cin >> ans;
        if (ans == "array"){
            test[0] = 1;
        }
        printf("2. It is an array that is capable of holding only one set of data: ");
        cin >> ans;
        if (ans == "one-dimensional"){
            test[1] = 1;
        }
        printf("3. It is an array that is capable of holding multiple sets of data on account of
having rows and columns: ");
        cin >> ans;
        if (ans == "two-dimensional"){
            test[2] = 1;
        }
        printf("4. It is two or more arrays that hold related data, and the related elements in
each array are accessed with a common subscript: ");
        cin >> ans;
        if (ans == "parallel"){
            test[3] = 1;
        }
        printf("5. It is an error that occurs when a loop does one iteration too many or too
little: ");
        cin >> ans;
        if (ans == "off-by-one-error"){
                test[4] = 1;
        }
        printf("\n\n\n");
        system("pause");
        int n = sizeof(test)/sizeof(test[0]);
   int x = 1;
        for (int i=0; i<n; i++)
     if (x == test[i])
       result++;
   cout <<"\nYou got : " << result << "/5\n";
        printf("This are the correct answers:\n");
        printf(" 1. array\n 2. one-dimensional\n 3. two-dimensional\n 4. parallel\n 5.
off-by-one-error\n");
        printf("\n\n\n");
        system("pause");
        while (sel != 1 || sel != 2){
                printf("\n1 - Back");
                printf("\n2 - Return to Lesson Menu");
                printf("\n\n\nSelect: ");
                scanf("%d", &sel);
                if (sel == 1)
                        L7();
                if (sel == 2)
                        menu();
        }
        printf("\n\n\n");
}
```

C. Work Breakdown

Student Name	Task Assigned	Percentage of the Work Contribution
BELANO, Paolo	Source Code Project Documentation - Review of Related Literature - Pseudocode - Results - Discussion of Results - User's Manual	33%
HERNANDEZ, Miro	Source code Project Documentation - IPO Charts - Hierarchy Chart - Flowcharts	33%
LIM, Ella Janelle	Program Content Project Documentation - Introduction - Pseudocode - Analysis, Conclusion, and Future Directives	33%

D. Personal Data Sheet

Surname: BELANO
Given Name: Paolo
Birthdate: December 11, 2001
Gender: Male
Occupation: Student
DLSU Email: paolo_a_belano@dlsu.edu.ph
Degree Program: BS in Computer Engineering



HERNANDEZ, Miro
Surname: Hernandez

Given name: Miro Manuel

Birthdate: November 30, 2001

Gender: Male

Occupation: Student

DLSU Email: miro_hernandez@dlsu.edu.ph

Degree Program: BS in Computer Engineering

Surname: Lim

Given name: Ella Janelle

Birthdate: August 18, 2001

Gender: Female

Occupation: Student

DLSU Email: ella_janelle_lim@dlsu.edu.ph

Degree Program: BS in Computer Engineering