Name:Deepanshi
Student #: 456807 Email ID:- d.807@mybvc.ca

SODV 1101

Programming Fundamentals

Winter 2024
Assignment 01

Instructor: Mahbub Murshed **Due Date:** 2024-02-26

Directions:

- Written questions can be submitted via soft copy or a scanned document and submitted by d2l dropbox, **as a single doc file or single pdf file**. Answers should be written clearly and coherently. **Programming questions must be submitted in repl.it.**
- Assignments must be submitted by the posted due date, or be subject to the course's late policy. Work must be completed individually and in accordance with the academic integrity policy.
- All codes shown in this assignment are compiled and tested except for questions that aim to find errors in a C++ program.
- For all written questions, you may assume that the usual header files are included as follows:

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;
```

Note: The assignment's written questions not submitted as a single file will not be graded!

Written Questions	
1	/ 5
2	/ 5
3	/ 5
4	/ 5
5	/ 5
6	/ 5
Programming Questions	
Quality	/ 20
Functionality	/ 50
Total	
	/ 100

Assignment 1: Written Questions

1. **Give concise answers to the following questions** / 5

a) **What is a compiler?** / 1

Ans: Compilers are specialized software tools that translate one programming language's source code into machine code, bytecode, or another programming language.
In simple words, compilers are programs that transform source code from a high-level programming language(human-friendly and easy to understand) to a lower-level language(machine-friendly and difficult to understand) in order to generate an executable program.

b) **What is pseudocode?** / 1

Ans: Pseudocode is an informal way of Programming description that does not require any strict programming language syntax or underlying technology considerations.
Pseudocode is a rough sketch to organize and understand a program before it is written in codes. It is more general as compared to algorithms as they are more specific.
Example:- Program that calculates the area of a square: (Pseudocode)

```
Get the length of one side

Multiply the length by itself

Store the result in a variable called area

Display the area
```

c) **What is the difference between an expression and a statement?** / 3

Ans:

S.No.	Expression	Statement
1	An expression usually refers to a piece of code that can be evaluated to a value, and is composed of variables, operators, function calls, and literals.	A statement refers to a piece of code that executes a specific instruction or tells the computer to complete a task
2	Expressions are method calls and mathematical operations. An expression will always yield an output; it evaluates to something that can be printed.	A statement can take the form of assignments, control statements, import statements, loop statements, jump statements, or method calls. Result from the evaluation cannot be printed.

3	Expressions are mostly defined by user.	Most statements are pre-defined in programming language.

4	**Example:-**
	
	5>3
	
	7*2+4
	In the example above, the first expression,5>3, will yield the value True when evaluated, while the second expression, 7*2+4, will yield the value 18.
	
	Evaluation based on rules of order of operations.
		Example:-
		
		x=20
		
		import library
		
		In the example above, the first statement performs the action of assigning the variable x with a value of 20. In the second statement, the instruction is given to import a library. Neither statement yields a value but instead performs a certain action.

2. **Give concise answers to the following questions** / 5

a) Illustrate the differences between syntax errors and logical errors with examples / 2

Ans:

Syntax Error	Logical Error
Any code or spelling and grammatical mistake that create any error during the coding or compiling.	Logical errors refer to the mistakes in the program's logic which gives incorrect output.
The program with syntax errors will not be executed by the compiler.	The program compiles and runs successfully even if it has logical error. Because of this error the computer represents the program, but the user is unable to find the real goal of programming.
Example:- return 0 //**missing semicolon** after return 0	Example:- int SquareArea(double width) /***Initializing** a function with **"int"** variable but accepting **"double"** is the **logical error***/

Example:- The below example show the exact difference between logical and syntax error.

```
#include<iostream>
using namespace std;
int SquareArea(double width) /*Initializing a function with "int" variable but accepting
"double" is the logical error*/
{
   return width * width;
}
```

```
int main()
{
double width;
   cout << "What is the width of the square? ";
   cin >> width;
   cout << "The area of the square is: " << SquareArea(width) << endl;
   return 0 //Syntax Error: missing semicolon after return 0
}
```

b) What will the function call q2(5, 5, 2) return? / 3

```
double q2(double x, int y, int z)
{
     x /= z;
     y /= z;
     return x + y;
}
```

Ans: The function will return the sum of x and y. So, 2.5 and 2 will give 4.5 as the output.
As, **x /= z;** This line divides x by z. So, x becomes 5 / 2, which is 2.5
y /= z; This line divides y by z. So, y becomes 5 / 2, which is 2 (both integer type)
return x + y; This line returns the sum of x and y. So, 2.5 + 2, which equals 4.5

Code Representation:-

```
#include<iostream>
using namespace std;

double q2(double x, int y, int z)
{
x /= z;
y /= z;
return x + y;
}
int main()
{
  double result = q2(5, 5, 2);
    cout << "Result: " << result << endl;
    return 0;
}
```

OUTPUT:-

```
Result: 4.5
```

3. Identify all the errors (syntax error and logical error) in the following program implementation Mark the errors in the code below with a circled number corresponding to the spaces provided, briefly explain why it is an error, and give a fix for it. /5

```
#include<iostream>
using namespace std;
/* SquareArea
 * @params: width - the width of the square
 * @returns: the area of the square
 */
int SquareArea(double width) Logical Error
{
    return width * width;
}

int main 1st Error
{
    cout << "What is the width of the square? ";
    cin << width; 2nd and 3rd error
    cout << "The area of the square is: " << SquareArea(width) << endl;
    return 0 4th error
}

//Note: The program code contains 4 syntax errors and a logical error.
```

1) Syntax Error: missing () after int main

2) Syntax Error: Expected cin >> width

3) Syntax Error: The variable width was not initialized inside int main()

4) Syntax Error: missing semicolon after return 0

5) Logic Error: Initializing a function with "int" variable but accepting "double.

CORRECT CODE:-

```
#include<iostream>
  using namespace std;
 double SquareArea(double width)
{
    return width * width;
}
int main()

{

   double width;

    cout << "What is the width of the square? ";

    cin >> width;

    cout << "The area of the square is: " << SquareArea(width) << endl;

return 0; }
```

4. **Give concise answers to the following questions** / 5

a) **Write truth tables for the logical AND, OR, and NOT operators.** / 3

Ans:

Truth Table		AND(&&) [If both operands are true, then the return value is true.]	OR(||) [If at least one operand is true, then the output is true.]
Operand 1	Operand 2		
FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	TRUE	TRUE	TRUE

Operand	NOT(!) [Returns the opposite of the operand]
FALSE	TRUE
TRUE	FALSE

b) What do you mean by a breakpoint? / 2

Ans: A breakpoint is a pause used for debugging purposes. It is one of the most important debugging techniques in developer's toolbox. We can set breakpoints wherever we want to pause debugger execution. Basically, it instructs the debugger to stop at a particular code location in any program so that the user can control the debugger.

/ 5

5.

a) What would be the output of the following code? And why? / 2.5

```
int main()
{
int x = 2;
switch (x)
{
        case 1: cout<<"Choice is 1";
        break;
        case 2: cout<<"Choice is 2";

        case 3: cout<<"Choice is 3";
        break;
        default: cout<<"Choice other than 1, 2 and 3";
        break;
}
return 0;
}
```

Ans: **Output:-** Choice is 2Choice is 3

The variable x is initialized with the value 2 and the switch statement evaluates the value of x since the x=2 matched with case 2 and the statement cout<<"Choice is 2"; is executed. However, there is no break statement after the cout statement in the case 2 so the control flow falls through to the next case label. The statement cout<<"Choice is 3"; is executed. It continues top execute until it reaches the end of the switch statement (break statement). Therefore, both case 2 and case 3 will be executed.

b) What will the function call q2b('R', 5, false); return from the following code? / 2.5

```
string q2b(char a, int b, bool c)
{
    string ret = "The ";
    if (b % 3 > 0)
        ret += "quick ";
    else
        ret += "slow ";
    c = (!c || c) && (a >= 'A' && a <= 'Z');
    if (c)
        ret += "brown ";
    else
        ret += "blue ";
    switch (b)
    {
    case 4:
        ret += "fox";
        break;
    case 5:
        ret += "penguin";
    case 6:
            ret += "bear";
            break;
        default:
            ret += "cat";
        }
        return ret;
}
```

Ans:- OUTPUT: The quick brown penguinbear

1) string ret =”The”; Initializes the string ret with the value “The”.
2) If **(b % 3 >0):** Since 5 % 3 equals 2, which is greater than 0, "quick " is appended to ret.
3) **c = (!c || c) && (a >= 'A' && a <= 'Z'); :** In this case, c is initially false. The condition (!c || c) && (a >= 'A' && a <= 'Z') evaluates to true && true because 'R' is indeed between 'A' and 'Z'. Therefore, c becomes true.
4) Since c is true, **"brown "** is appended to ret.
5) **switch (b):** b is 5, so it matches the case 5. "penguin" is appended to ret. However, there is no break statement after "penguin", so the control falls through to the next case.
6) **"bear"** is appended to ret.
7) The function returns the string ret.
8) So, the function call **q2b('R', 5, false);** will return the string **"The quick brown penguinbear"**.

6. In the following function implementation, there are 4 syntax errors and 1 type of logical error [same error type can be there in one or more lines]. Mark the errors in the code below with a circled number corresponding to the spaces provided. Briefly explain why it is an error and give a fix for it. / 5

```
#include<iostream>
using namespace std;
/* Calculator
 * @params: command - the operation you want to perform (+, -, *, or /)
 *          op1, op2 - the 2 operands
 * @returns: the calculated answer
 */
int Calculator(char command, int op1, int op2);     1
{
    answer = 0; 2
    switch command   3
    {
    case '+':
        answer = op1 + op2;    5
    case '-':
        answer = op1 - op2;    5
    case '*':
        answer = op1 * op2;    5
    case '/':
        answer = op1 / op2;    5

    }
    return answer   4
}
```

1) Syntax Error: int Calculator(char command, int op1, int op2);
Unexpected semicolon(error)

2) Syntax Error: answer = 0; ‘answer’ is not declared

3) Syntax Error: switch command
Missing parentheses for switch condition

4) Syntax Error: return answer
Missing Semicolon

5) Logic Error: The absence of **break statements** within the switch cases. Without break statements, the **control will fall through to subsequent cases even after one case is matched.**
If I want to display the Case ‘+’ but due to absence of break statement we get the output of case ‘/’ like op1 is 2 and op2 is 1 and want my console display the addition of two number ie 3 but due to absence of break statement we get 2

Corrected Code:-

```
#include <iostream>
using namespace std;

int Calculator(char command, int op1, int op2) { // Function definition
    int answer = 0; // Declare and initialize 'answer'

    switch (command) { // Switch statement to determine the operation
        case '+':
            answer = op1 + op2; // Addition
            break;
        case '-':
            answer = op1 - op2; // Subtraction
            break;
        case '*':
            answer = op1 * op2; // Multiplication
            break;
        case '/':
                answer = op1 / op2; // Division
            break;
    }

    return answer;
}

int main() {
    char command;
    cout<<"The selected operation is:";
    cin>>command;
    int operand1 = 10; //   Pre-defined First operand
    int operand2 = 5; //    Pre-defined Second operand
    cout << operand1 << " " << command << " " << operand2 << " = " <<
Calculator(command, operand1, operand2) << endl;

    return 0;
}
```

REFERENCES

Wikimedia Foundation.(n.d.) Compiler. *Wikipedia.* Retrieved February 2, 2024, from https://en.wikipedia.org/wiki/Compiler

The India Times.(n.d.) Pseudocode. *EconomicTimes.* Retrieved February 25, 2024, from https://economictimes.indiatimes.com/definition/pseudocode

Baeldung.(July19,2021). Experssion-vs-Statement. *BaeldungCS*, from https://www.baeldung.com/cs/expression-vs-statement

Microsoft.(December 06, 2023). Use breakpoints in the Visual Studio debugger. *Microsoft VisualStudio,* from https://learn.microsoft.com/en-us/visualstudio/debugger/using-breakpoints?view=vs-2022

Program A: BMI Calculator (Please submit in Repl.it)

Learning Objective

- Identify program structure and control the flow of execution.
- Define variables and use them to store data.
- Write structured programming instruction so that the computer shows input/output behaviors
- Use operators to perform calculations.
- Employ data store elements in program implementation.
- Demonstrate logical thinking by using programming syntax and strategies.

Overview

Write a program to calculate the user's body mass index (BMI).

Directions

A person's body mass index (BMI) is calculated using height and weight. The formula for BMI is:

BMI = kg / m^2

Where kg is the person's mass in kilograms, and m^2 is their height in meters squared.

Write a program to calculate a person's BMI. Your program should first ask the user to enter their height in feet and inches, then their weight in pounds. Your program should then use the data to calculate the person's BMI. Your program should also display the person's height and weight in meters and kilograms. There are 12 inches in a foot, 3.28084 feet in a meter, and 2.20462 pounds in a kilogram.

The listing below shows an example of what the input and output for this program might look like. Input by the user is listed in **blue**.

```
Sample Input/Output:

What is your height?
Feets: 5
Inches: 10
What is your weight (lbs)? 160

Height (m): 1.78
Weight (kg): 72.57
BMI: 22.96
```

Submitted on replit

Program B: Restaurant Bill (Please submit in Repl.it)

Learning Objective

- Identify program structure and control the flow of execution.
- Define variables and use them to store data.
- Write structured programming instruction so that the computer shows input/output behaviors
- Use operators to perform calculations.
- Describe operator precedence and expressions.
- Employ data store elements in program implementation.
- Demonstrate logical thinking by using programming syntax and strategies.

Overview

Write a program to split a restaurant bill evenly between a given number of people.

Directions

Write a program for calculating and splitting restaurant bills. Your program should ask the user how much the bill is before applying tax, what percentage the sales tax is, what percentage tip they would like to add, and how many people are splitting the bill. Calculate the final cost of the bill by first adding the sales tax, then adding the tip. Find the cost per person by dividing the bill by the number provided. Print each value to the console: the bill before tax, the bill after tax, the bill after tip, and the bill per person.

The listing below shows an example of what the input and output for this program might look like. Input by the user is listed in **blue**.

```
Sample Input/Output:

How much is the bill? 57.75
How much is the tax (%)? 5
How much is the tip (%)? 20
How many people are paying? 3

Bill before tax: 57.75
Bill after tax:  60.64
Bill after tip:  72.76
Bill per person: 24.26
```

Submitted on replit

Program C: Monthly Telephone Bill (submit in Repl.it)

Learning Objective

- Identify program structure and control the flow of execution.
- Define variables and use them to store data.
- Write structured programming instruction so that the computer shows input/output behaviors
- Use operators to perform calculations.
- Describe operator precedence and expressions.
- Employ data store elements in program implementation.
- Create basic functions to reuse code.
- Demonstrate logical thinking by using programming syntax and strategies.

Overview

Write a program to calculate the monthly telephone bills as per the following rule:

Directions

You must accept the total number of calls from the user and calculate the monthly (pay as you go) bill according to the following rule.

Minimum $20 for up to the first 100 calls.
Plus $0.30 per call for the next 50 calls.
Plus $0.20 per call for the next 50 calls.
Plus $0.10 per call for any call beyond 200 calls.

Sample Input/Output 01:

```
Enter the total number of calls: 130
Your Monthly bill is:  $29
```

Sample Input/Output 02:

```
Enter the total number of calls: 180
Your Monthly bill is:  $41
```

Sample Input/Output 03:

```
Enter the total number of calls: 500
Your Monthly bill is:  $75
```
(Submitted on replit)

Evaluation

This assignment has 2 main components, assigned marks as follows:

Task	Marks
Written Questions	30
Code Quality and Functionality	70
Total	100

Marks for written questions are indicated beside each question.

The **following rubrics will be followed for assessing the program code:**

Excellent (100%)	*Competent (80%)*	*Satisfactory (50%)*	*Unsatisfactory (0%)*
• *Code contains no compile-time errors.* • *Code contains no logical errors.* • *Code produces the exact correct output and does not crash for valid inputs.* • *Code is well organized, clear, and easy to read.* • *Code is consistent throughout and comments are used where needed.*	• *Code contains no compile-time errors.* • *Code produces the exact correct output except for a few formatting issues / incorrect data type issues.* • *Code does not crash for valid inputs.* • *Code produces the correct output most of the time. However, for some valid inputs, it shows the incorrect output.* • *Minor logical errors are present.*	• *Code contains no compile-time errors.* • *Code is disorganized or could use some refactoring.* • *Code contains major logical errors and does not match the expected output.*	• *Code contains compile-time errors.* • *Code contains major logical errors.* • *Code is very difficult to read.* • *Code does not execute.*