CMIS 102 Hands-On Lab

Week 5

Overview

This hands-on lab allows you to follow and experiment with the critical steps of developing a program including the program description, analysis, test plan, design (using pseudocode), and implementation with C code. The example provided uses sequential, selection and repetition statements.

Program Description

This program will calculate the average of 10 positive integers. The program will ask the user to enter 10 integers. If any of the values entered is negative, a message will be displayed asking the user to enter a value greater than 0, and the program will ignore the negative input and ask for a valid value. The program will use a loop to input the data.

Note that the program will always expect 10 *positive* integers to be entered. If a negative number is entered, the user needs to enter a positive number to replace it.

Analysis

I will use sequential, selection and repetition programming statements.
I will define three integer variables: count, value and sum. count will store how many times values are entered. value will store the input. sum will store the sum of all 10 integers.
I will define one double variable: avg. avg will store the average of the ten positive integers input.

The sum will be calculated by this formula:
  sum = sum + value
For example, if the first value entered was 4 and second was 10:
  sum = sum + value = 0 + 4
  sum = 4 + 10 = 14
Values and sum can be input and calculated within a repetition loop:
while count < 10
   Input value
   sum = sum + value
End while

Avg can be calculated by:
  avg = sum / count

A selection statement can be used inside the loop to make sure the input value is positive, and that only positive values are included in the sum.

```
If value >= 0 then
   count = count + 1
   sum = sum + value
Else
   input value
End If
```

Test Plan

To verify this program is working properly the input values could be used for testing:

Test Case	Input	Expected Output
1	value=1 value=1 value=1 value=0 value=1 value=2 value=0 value=1 value=3 value=2	average is 1.2
2	value=100 value=100 value=100 value=100 value=100 value=200 value=200 value=200 value=200 value=200	average is 150.0
3	value=100 value=100 value=100 value=100 value= -100 value= 100 value=200 value=200 value=200 value=200 value=200	Value must be positive average is 150.0

Pseudocode

```
// This program will calculate the average of 10 positive integers.

// Declare variables
Declare count, value, sum as Integer
Declare avg as double

//Initialize value
```

```
Set count=0
Set sum = 0

// Loop through 10 integers
While count < 10
   Print “Enter a Positive Integer”
   Input value
   if (value >=0)
      sum = sum + value
      count=count+1
   else
      Print (“Value must be positive”)
   End if
End While

// Calculate average
avg = sum/count
// Print results
Print “Average is “ + avg
```

C Code

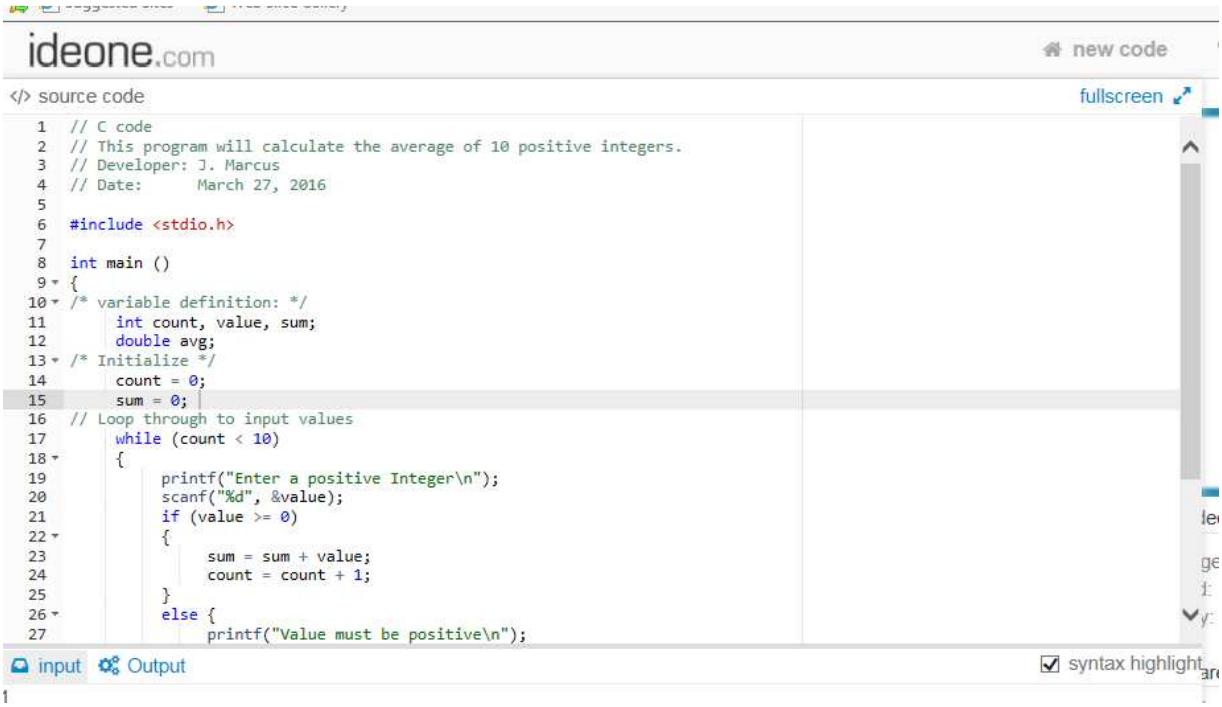
The following is the C Code that will compile and execute in the online compilers.

```
// C code
// This program will calculate the average of 10 positive integers.
// Developer: PUT IN YOUR NAME
// Date: PUT IN DATE PROGRAM DONE

#include <stdio.h>

int main ()
{
/* variable definition: */
     int count, value, sum;
     double avg;
/* Initialize */
     count = 0;
     sum = 0;
// Loop through to input values
     while (count < 10)
     {
          printf("Enter a positive Integer\n");
          scanf("%d", &value);
          if (value >= 0)
          {
               sum = sum + value;
               count = count + 1;
          }
          else {
               printf("Value must be positive\n");
          }
     }
```

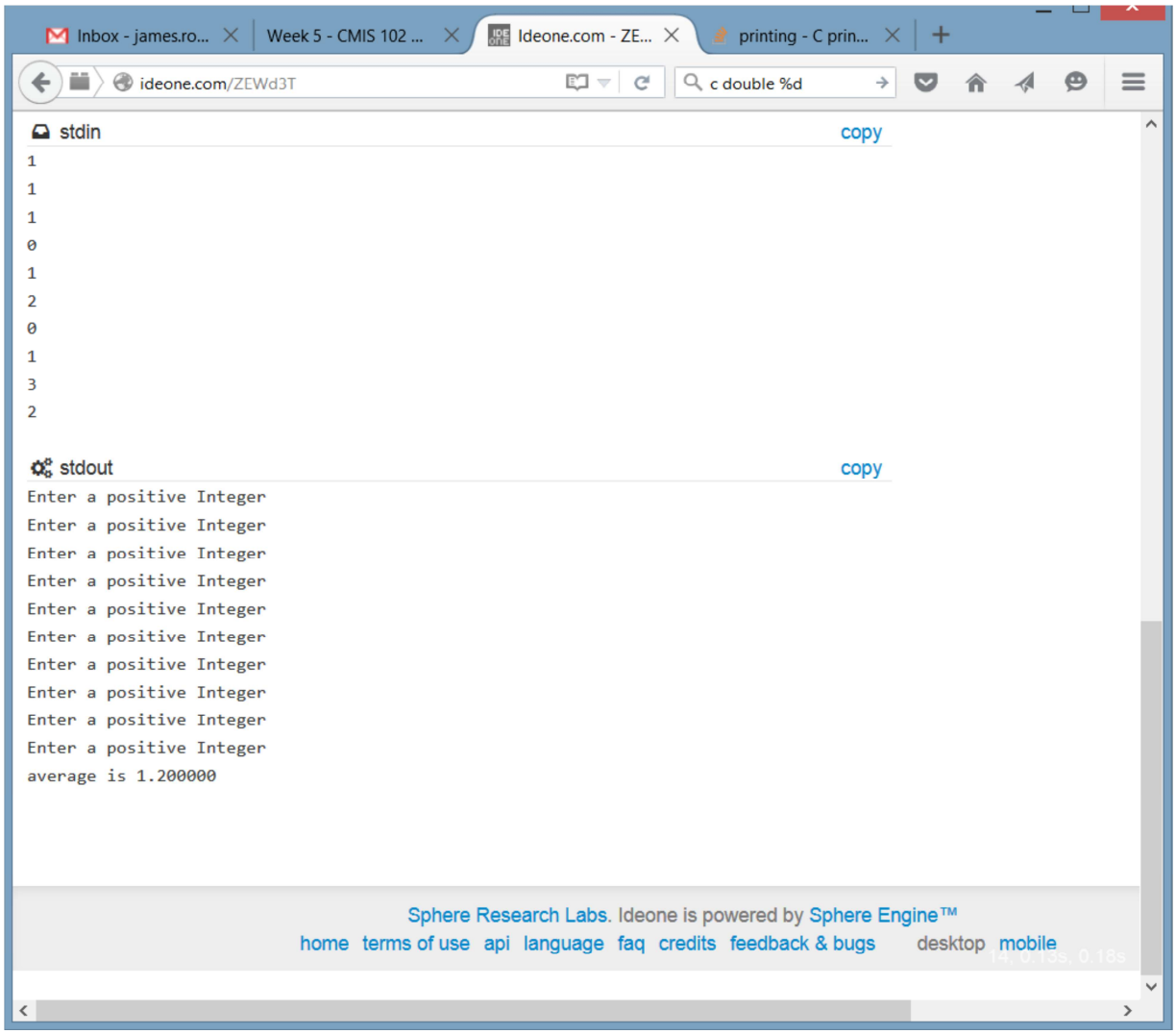
```
// Calculate avg. Need to type cast since two integers will yield an integer

     avg = (double) sum / count;
     printf("average is %lf\n " , avg );
     return 0;
}
```

Setting up the code and the input parameters in ideone.com:

Note the input integer values are 1, 1, 1, 0, 1, 2, 0, 1, 3, 2 for this test case. You can change these values to any valid integer values to match your test cases. You should also test with a negative number to make sure the positive integer logic works properly.



Results from running the programming at ideone.com



Learning Exercises for you to complete .

1. Load the baseline program into a compiler, compile it and run it. Provide a full program listing (including all comments), and a screen capture showing the input data and the program executing.
2. Change the code to average 20 integers as opposed to 10. Support your experimentation with a full program listing, and screen captures of the input data and the new code executing.
3. Prepare a new test table with at least 3 distinct test cases listing input and expected output for the code you created in step 2. Be sure that your test data checks every logic path through the program. Remember that if you enter a negative input, you need an additional input to make up for the erroneous input.
4. What happens if you entered a value other than an integer? (For example a float or even a string). You will need to modify your code to put in debugging statements that show what is happening inside the program when erroneous inputs are made. In addition, your test data should make it easy to tell which input is being made. Support your experimentation with screen captures of executing the code with the debugging statements.
5. Modify the code to allow the user to enter an unspecified number of positive integers and calculate the average. In other words, the user could enter any number of positive integers. You should *not* ask the user how many inputs they will have. The program will need to determine how many numbers have been entered. (Hint: You can use a sentinel value to trigger when the user has completed entering values. As an alternative, you may use an EOF loop). Prepare a new test table with at least 3 distinct test cases listing input and expected output for the code you created. If you use a sentinel loop, be sure to include the sentinel value. Support your experimentation with a full program listing, including all comments, and screen captures showing the input data and the new code executing.

Grading guidelines

Submission	**Points**
Successfully demonstrates execution of this lab with online compiler. Includes a program listing and a screen capture showing the program executing.	2
Modifies the C code to average 20 integers as opposed to 10. Provide a full program listing (including comments) and screen captures showing the revised code executing.	2
Provides a new test table with at least 3 distinct test cases listing input and expected output for the code you created in step 2.	1
Describes what happens if you entered a value other than an integer. Support your experimentation with screen captures of executing the code.	1
Modifies the C code to allow the user to enter an unspecified number of positive integers and calculate the average. Provides a new test table with at least 3 distinct test cases listing input and expected output for the code you created. Provides a listing of the full program, including comments, and a screen captures showing the new code executing.	3
Document is well-organized, and contains minimal spelling and grammatical errors.	1
Total	**10**