INTRODUCTION TO PROGRAMMING
DCS20103

SEPTEMBER 2022

LAB PRACTICAL 2

		
ADRIAN DWANE BILLONES	LAZARUS THOMAS	TENGKU PUTERI NUR'AIN ANTASYA THURAYA BINTI TENGKU HISHAMMUDDIN ZAIZI
012022070189	012022070163	012022070704
BACHELORS IN GAMES DESIGN AND ANIMATION	BACHELORS IN GAMES DESIGN AND ANIMATION	BACHELORS IN GAMES DESIGN AND ANIMATION

Lecturer: MADAM NOR KAMALIAH BINTI KHAMIS
Date of Submission :

Marks:

REFER RUBRIC	**Marks**
Question 1	
Question 2	
TOTAL	**/100**

Table of Contents

QUESTION 1

Write a program that reads an unspecified number of integers, determine how many positive and negative values have been read, and computes the total and average of the input values (not counting zeros). Your program ends with the input 0. Display the average as floating-point number.

Sample of output:

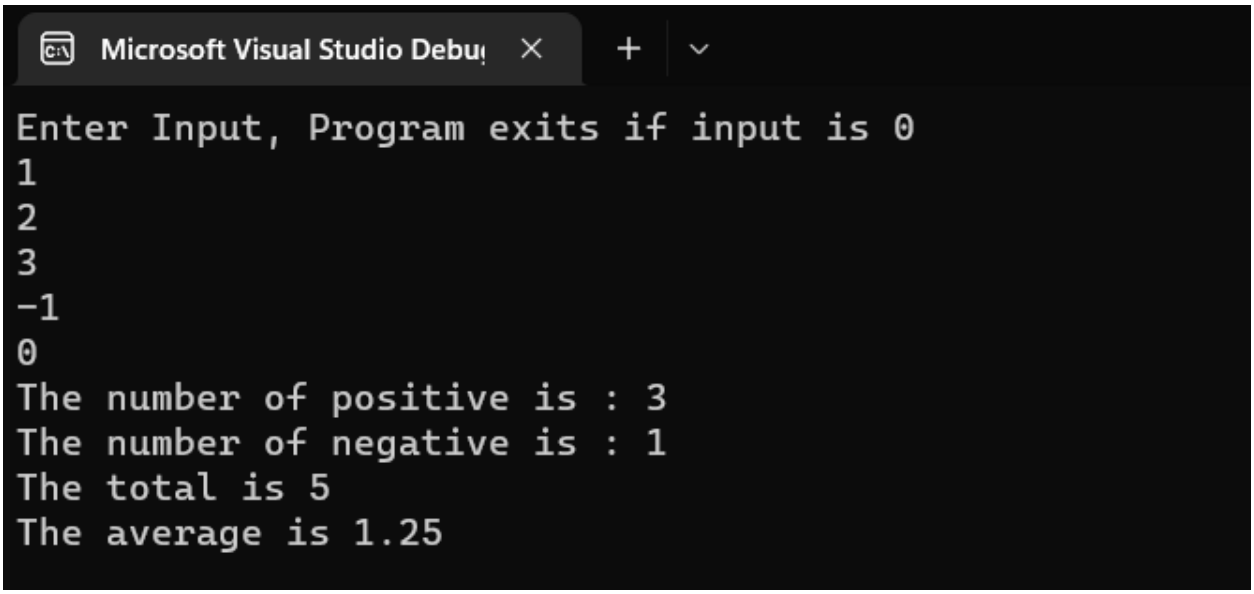
```
Enter an int value, the program exits if the input is 0:
        2  -1  3  0

The number of positive is 3
The number of negative is 1
The total is 5
The average is 1.25
```

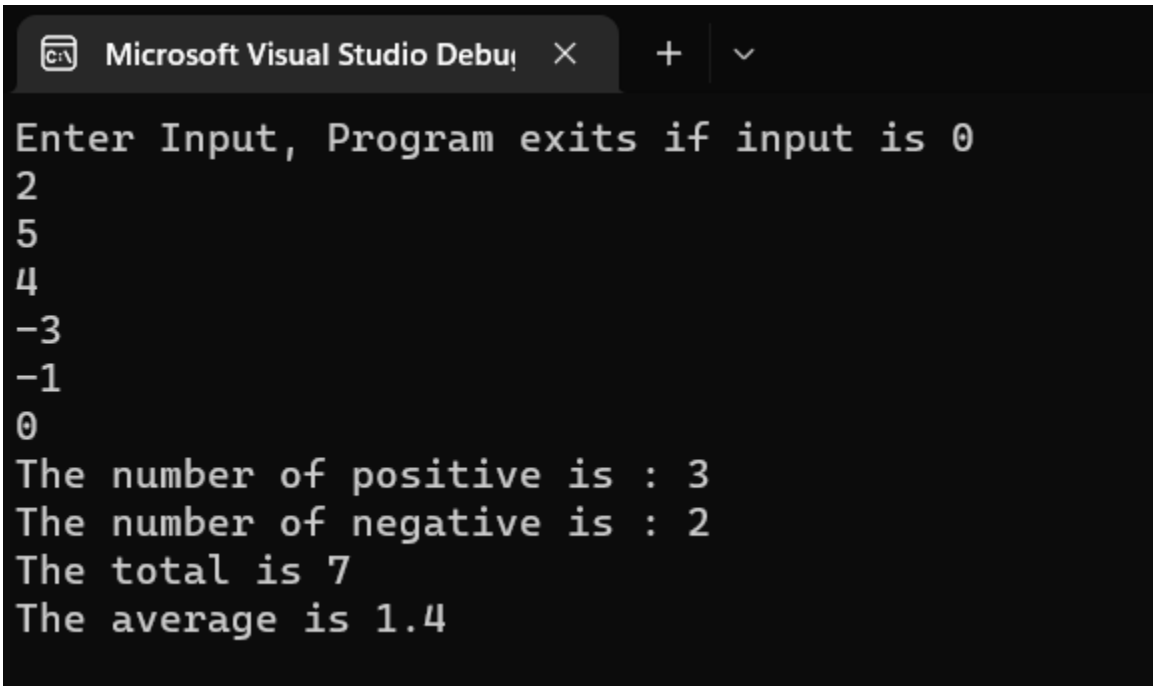
QUESTION 1 ANSWER

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

int main()
{
// assigning all variables
    int  arr[100], num, negativenum = 0, positivenum = 0;
    int i = 0;
    int count = 0;
    float sum = 0.0, avg;

//user input numbers
    cout << "Enter Input, Program exits if input is 0" << endl;

    while (1)
    {
// if enter 0, the loop will break and program will stop input
        cin >> num;
        arr[i] = num;
        if (num == 0)
        {
            break;
        }

// if else statement to categorize which number is positive or negative
        else if (num > 0)
        {
            positivenum++;
```

```
            sum += num;
        }
        else
        {
            negativenum++;
            sum += num;
        }
        i++;
    }
//count used to calculate total numbers input. count variable will be used to find average with given sum of input values
    count = positivenum + negativenum;
// formula for finding the average
    avg = sum / count;

    {

// final display
        cout << "The number of positive is : " << positivenum << endl;
        cout << "The number of negative is : " << negativenum << endl;
        cout << "The total is " << sum << endl;
        cout << "The average is " << avg << endl;



        return 0;

    }

}
```

OUTPUT


Microsoft Visual Studio Debu
Enter Input, Program exits if input is 0
1
2
3
-1
0
The number of positive is : 3
The number of negative is : 1
The total is 5
The average is 1.25


Microsoft Visual Studio Debu
Enter Input, Program exits if input is 0
2
5
4
-3
-1
0
The number of positive is : 3
The number of negative is : 2
The total is 7
The average is 1.4

QUESTION 2

Write a program that assigns four subjects which are Mathematic, Chemistry, Biology and Physic to an array name ***subject*.** Then, the program will ask the user to enter marks for each subject. Find the average of the total marks and display the subject which is the highest and the lowest marks.

Sample of output:

Mathematic : **95**
Chemistry : **85**
Biology : **70**
Physic : **90**

Average of the total marks : **80**
Subject with highest marks : **Mathematic**
Subject with lowest marks : **Biology**

Do you want to continue? Press Y or y
y

The program also asks the user to decide whether he/she wants to continue the operation. If he/she inputs 'y', the program will prompt the user to enter the marks again. Otherwise, the program will terminate.

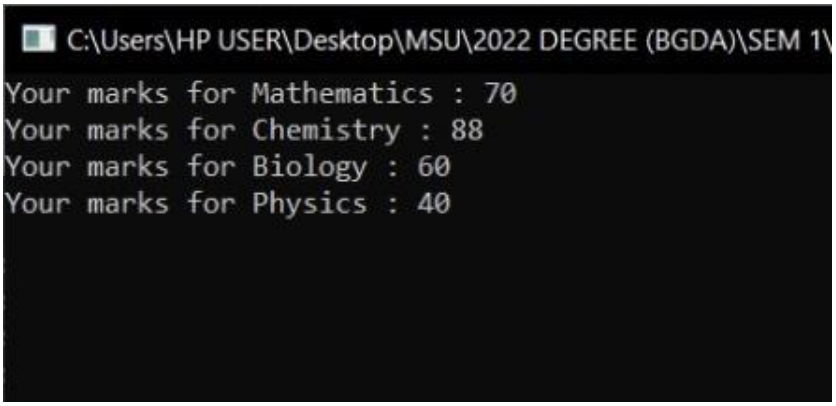
QUESTION 2 ANSWER

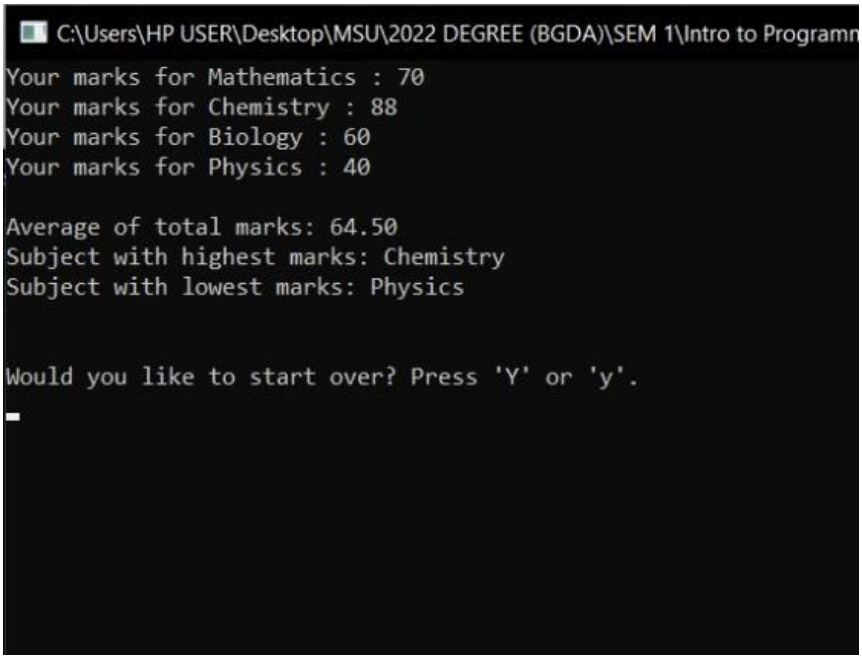
```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

int main()
{
	char ans;
	const int SUBJ=4;
	string name[SUBJ]={"Mathematics","Chemistry","Biology","Physics"};
	double marks[SUBJ], avg, max, min;
	string highMark, lowMark;

	do{
	//ask user for marks for each subject
	int i=0;

	for (int i=0; i<SUBJ; i++)
	{
		cout<<"Your marks for "<<name[i]<<" : ";
		cin>>marks[i];
	}

	//avg marks calculation
	double totalMarks=0;

	for (int i=0; i<SUBJ; i++)
	{
		totalMarks+=marks[i];
		avg=totalMarks/SUBJ;
	}
```

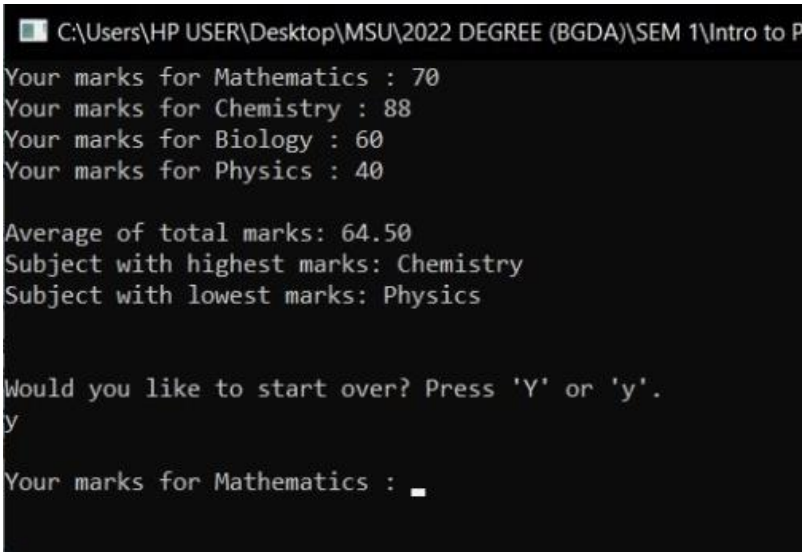
```
//subjects and their respective marks
for (int i; i<SUBJ;i++)
{
	cout<<name[i]<<" : "<<marks[i]<<endl;
}

//subject w highest mark
max=marks[0];

for (int i=1; i<SUBJ;i++)
{
	if (marks[i]>max)
	{
		highMark=name[i];
		max=marks[i];
	}
}

//subject w lowest mark
min=marks[0];

for (int i=1; i<SUBJ; i++)
{
	if (marks[i]<min)
	{
		lowMark=name[i];
		min=marks[i];
	}
}

cout<<endl;
```

```
        cout<<setprecision(2)<<fixed<<"Average of total marks: "<<avg<<endl;
        cout<<"Subject with highest marks: "<<highMark<<endl;
        cout<<"Subject with lowest marks: "<<lowMark<<endl;
        cout<<endl;
        cout<<endl;
        cout<<"Would you like to start over? Press 'Y' or 'y'."<<endl;
        cin>>ans;
        cout<<endl;
        }while (ans=='y'||ans=='Y');
}
```

OUTPUT




C:\Users\HP USER\Desktop\MSU\2022 DEGREE (BGDA)\SEM 1\Intro to Programm
Your marks for Mathematics : 70
Your marks for Chemistry : 88
Your marks for Biology : 60
Your marks for Physics : 40
Average of total marks: 64.50
Subject with highest marks: Chemistry
Subject with lowest marks: Physics
Would you like to start over? Press 'Y' or 'y'.


C:\Users\HP USER\Desktop\MSU\2022 DEGREE (BGDA)\SEM 1\Intro to P
Your marks for Mathematics : 70
Your marks for Chemistry : 88
Your marks for Biology : 60
Your marks for Physics : 40
Average of total marks: 64.50
Subject with highest marks: Chemistry
Subject with lowest marks: Physics
Would you like to start over? Press 'Y' or 'y'.
y
Your marks for Mathematics :

Programming Rubric

Criteria	Satisfactory (2)	Good (3)	Excellent (4)	Q1	Q2
Delivery	• Completed, but fail to submit on time. Assignment up to 3 days late or turned in incorrectly.	•Completed, but fail to submit on time. Assignment up to one day late but otherwise turned in correctly.	• Completed and submit on time and neatly with all sections clearly labeled and stapled together in the correct order. (5 marks)		
Naming	•Names / variables/ constant are not consistent, and occasionally verbose, overly terse, ambiguous or misleading.	• Completed, but fail to submit on time. Assignment up to one day late but otherwise turned in correctly.	• All names / variables/ constant are consistent with regard to style and are expressive without being verbose.		
Readability, Consistency & Coding Standards	• Coding style guidelines are not followed and/or code is less readable than it should be. •Fair in organizing the work. •Not use manipulator etc (if required)	• Coding style guidelines are almost always followed correctly. Code is easy to read. • Good in organizing the work. • Use manipulator etc in certain way (if required)	• Coding style guidelines are followed correctly, code is exceptionally easy to read and maintain. • Excellent in organizing the work. • Use manipulator etc effectively (if required)		
Design, Structure & Efficiency	•Program isn't as clear or logical as it should be. •Control structures /array/ function are occasionally used incorrectly.(if any) •Steps that are clearly inefficient are used. •A logical solution that is easy to follow but it is not the most efficient.	• Program is mostly clear and logical. • Control structures /array/ function are used correctly. (if any) • Reasonable algorithms are implemented • Solution is efficient and easy to follow (i.e. no confusing tricks).	• Program is designed in a clear and logical manner. • Control structures /array/ function are used correctly.(if any) • The most appropriate algorithms are implemented. • Solution is efficient, easy to understand, and maintain.		
User Interface/ Prompt Line	•Screen based instructions and final output are not clear, are not correct or are not attractive. •And/or Program is not "user friendly.	• Screen based instructions and final output are mostly clear, correct and attractive. • Program is "user friendly" with informative and consistent prompts and messages.	• Screen based instructions and final output are clear, correct and attractive. • Program is "user friendly" with informative and consistent prompts and messages.		
Correctness Testing & Output	•Program approaches correct answers or appropriate results for most inputs, but can contain miscalculations in some cases. •Testing is complete and there were redundant. •Nearly all boundary cases are considered and tested. •No sample of output provided. **•0 mark if incorrect output**	• Program produces correct answers and appropriate results for most inputs. • All key items are tested, but testing may be redundant. Nearly all boundary cases are considered and tested. • Provide samples of output.	• Program produces correct answers and appropriate results for all inputs tested. • Testing is complete without being redundant. All boundary cases are considered and tested. • Provide 2 samples of correct output.		
			TOTAL (25 for each question)		