National University of Computer and Emerging Sciences, Lahore Campus

Course:	PF Lab	Course Code:	CL-118
Program:	BS (Computer Science)	Semester:	Fall 2019
Duration:	150 Minutes	Total Marks:	60(30+30)
Paper Date:	28 Nov 2019	Weight	40%
Section:	A, B, G, H, I, J	Page(s):	2
Exam:	Final Term	Reg. No	

Instruction/Notes:

1. Understanding the question paper is also part of the exam, so do not ask any clarification.
2. No USB's, PHONES and INTERNET are allowed.
3. Talking/Discussion is not allowed. It is your responsibility to protect your code and save it from being copied. If you don't protect it all matching codes are considered copy/cheating cases.
4. Create a folder as you roll number. e.g **l191234** and put both source (.cpp files) inside folder and submit on following submission path: **\\cactus\Xeon\Fall 2019\Shakeel Zafar\PF Final Exam # 2\Section X**. Where X is your section.

Question # 1: [Empirical-Rule of Data]

You are given with some real data. You need to check whether the data make good bell shape or bad bell shape. This goodness or badness is computed through a rule. We name that rule as **Empirical-Rule of Data**. The rule says at least 65% and at most 75% values lie with in 1st standard deviation. How standard deviation is calculated?

Here is the Standard Deviation formula:

S = standard deviation
N = total values
X_i = a single value from sample data
$\overline{X}$ = the average value of the data

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}$$

So, what is the meaning of *"values lie with in 1st standard deviation"*?

For that you need to calculate the **count** of values which are bigger than ($\overline{X}$ - S) and lesser than ($\overline{X}$ + S). This **count** helps to calculate the **percentage**. The **percentage** is calculated through (**count** / **N**) *100. So, if this percentage is between 65 and 75, then it follows the **Empirical-Rule of Data,** and the data makes good bell shape**.**

You need to implement these functions in order to meet the given sample output.

```
float getStandardDeviation (float arr [], int size);              //returns the standard deviation
float getAverage (float arr [], int size);                        //returns the average value
void inputArray (float arr [], int size);                          // input array from (0-99)
float percentage (float arr [], int size, float average, float std); //returns the percentage
```

You are given with a main function and a sample input/output.

Sample Output:

Sample Data: 42, 34, 69, 79, 6, 82, 6, 95, 61, 28
Average of Data: 55.8
Standard Deviation: 26.3051
The percentage of data with in 1st standard deviation is: 70%
Good Bell shape

```
void main() {
    float data[10]; int size = 10;
    inputArray(data, size);
    printData(data, size);
    float average = getAverage(data, size);
    float std = getStandardDeviation(data, size);
    cout << average << endl << std << endl;
    float per = percentage(data, size, average, std);
    cout << per << endl;
    if (per > 65 && per < 75)
        cout << "Good Bell shape\n";
    else
        cout << "Bad Bell shape\n";
}
```

Question # 2: [Removal of Duplicate Words]

You need to write a c++ program which have a character array initialized with some data. Then you have to remove all the duplicate words from the data and print the updated data on console.

You need to implement these functions in order to meet the given sample output.

```
void removeDuplicateWords (char arr []);
```

This function receives the array and removes duplicate words. This function makes a new character array to store a word/substring. Every word in arr is separated by a single space. When a word/substring is formed, it calls the function findSubString with the array, substring and an index value, from where the search should begin.

```
int findSubString (char arr [], char sub [], int index);
```

This function receives an array of characters and a substring to be searched and an index value, from where the search will start. This function returns the index where the substring is found else it returns -1.

Note: Each word is separated by a single space. For your ease, you can make a new array in the **`removeDuplicateWords`** function.

```
void main() {
    char arr[] = "Hello I am muslim I am going to Lahore Hello";
    removeDuplicateWords(arr);
    cout << arr << endl;
}
```

Sample I/O:

arr:	Hello I am muslim I am going to Lahore Hello
Output:	Hello I am muslim going to Lahore

---------- **BEST OF LUCK** 😊 ----------