Lahore University of Management and Sciences

Lab-6 (Midterm) Manual

for

Introduction to Programming

(CS200)

Dr. Mian Muhammad Awais

LAB GUIDELINES

- Make sure you start uploading the lab before 11:50 AM. The tab on LMS for Midterm will be taking submissions till 12:15pm ONLY and any late submission will not be graded.
- **You should email your assigned TA ONLY**.
- For every lab, there will be a folder created on LMS. You must submit your work in the respective folder during the lab time, you and only you are responsible for your submissions.
- You will be allowed to discuss the questions in the first half of the lab session for a few questions. After that, there will be a portion of lab where you cannot converse and must work for yourself. No discussion is allowed in later time period.
- You should do your work with utmost clarity and precision. Do not waste your time trying to do something you do not understand. Ask Lab instructors for help, that is what they are there for.
- Any legitimate cheating case can and will be reported to Disciplinary Committee without any leniency. Plagiarism Software make our task easier.
- Please follow the lab etiquettes and follow code of conduct in the session.

- Do not start Personal chat during your zoom meeting and raise your hands before asking questions.
- **Make separate .cpp files for each question. Naming convention for .cpp files is: YourRollNumber_TaskX.cpp. Before submission, copy all the .cpp files in a folder named LabX_AssignedTAName_YourRollNumber.zip. Submit the .zip file only! (no .rar file submissions). X should be replaced by appropriate number.** Failure to follow the naming convention may lead to deduction of marks.

OBJECTIVES

- **Testing all concepts taught in class till now**

LAB EXERCISES

Question # 1 [Marks: 20] Est. Time: 30 mins

In this question you are required to make a struct "**memer**" and a class "**meme**". Data members for meme class will be:

- **string caption**
- **string image** (Input should be a simple string but must be stored in this variable with .jpg extension for example if input for image name = "CS200", then "CS200.jpg" should be stored as image name)
- **string author**

NOTE: Do NOT enter .jpg at the end of file name. It should be added implicitly by your program. Since all of you are not familiar with getline() yet, make sure name, caption and author do not contain any spaces

You also have to implement the following function for meme class:

- **default and parametrized constructors**
- **setter and getters**
- **displayMeme** (prints all the members in proper format as shown in sample output)

For memer struct, you have one data member **string name** and one member function that does the following:

- **makeMeme** (it takes pointer of meme object as its parameter and sets its members caption and image after prompting them from user. Author of meme object should be set to the name of the memer)

In int main(), you have to declare one object of meme and one of memer. Hard code **name** of memer to your name (no spaces!). Make a call to makeMeme essentially creating a meme. Once done, call displayMeme to see your meme!

Sample Output # 1

```
Enter caption for your meme: ZoomUniveristy
Enter image name for your meme (do NOT include the extension .jpg): ZoomLogo
Caption: ZoomUniveristy
Image: ZoomLogo.jpg
Author: SIMSB.

Press any key to continue . . .
```

Marks' distribution:

Proper private and public assignment: 5

Proper skeleton for the struct and the class: 10 (5 for each)

Correct output with proper formatting: 5

Question # 2 [Marks: 30] Est. Time: 40 mins

In this task, you need to sort the integer array and later perform arithmetic operations on the elements in the sorted array. You are not required to make any struct/class.

In int main() module, prompt the user for the size of an integer array as well as its elements. After getting the input, pass this array to a function, called **sortArray()** whose sample prototype is given to you below (you MUST follow the prototype as is without making any changes to it). In this function, you need to sort the array in ascending order and then come back to the main() module, returning nothing

void sortArray(int* array, int size);

Afterwards, you need to calculate the mean, median, mode and range of the array elements (integers). [You must take care of the cases where there is no mode in the array / more than one. In case of no mode, consider array[0] as the mode. In case of more than one modes, consider the

lowest value with maximum occurrence as the mode of the array. Refer to sample output for clarification]

Median:

- **{(n+1)/2}th term (if n is odd)**
- **[(n/2)th term + {(n/2)+1}th]/2 (if n is even)**

Range = Highest value – Lowest value

Mode is the most occurring value in the array

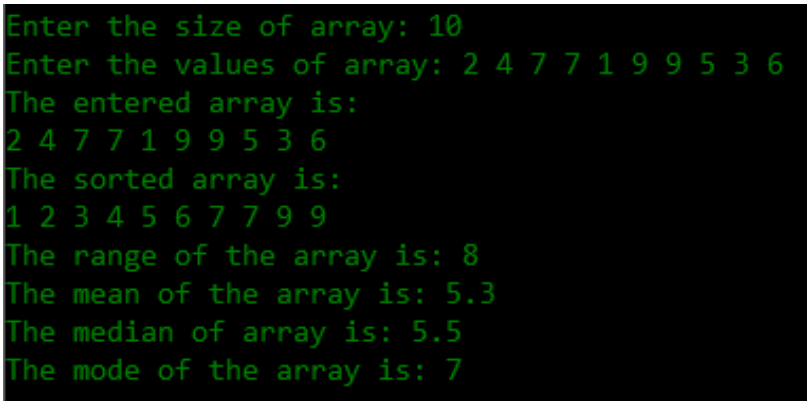
Sample Output # 1

```
Enter the size of array: 8
Enter the values of array: 1 2 3 7 6 5 9 3
The entered array is:
1 2 3 7 6 5 9 3
The sorted array is:
1 2 3 3 5 6 7 9
The range of the array is: 8
The mean of the array is: 4.5
The median of array is: 4
The mode of the array is: 3
```

Sample Output # 2 (In case of no mode)

```
Enter the size of array: 10
Enter the values of array: 8 1 2 6 5 3 4 7 9 0
The entered array is:
8 1 2 6 5 3 4 7 9 0
The sorted array is:
0 1 2 3 4 5 6 7 8 9
The range of the array is: 9
The mean of the array is: 4.5
The median of array is: 4.5
The mode of the array is: 0
```

Sample Output # 3 (In case of more than one mode)



Marks' Distribution:

Sort: 8

Mean: 4

Median: 4

Mode: 8

Range: 2

Output: 4

Question # 3 [Marks: 50] Est. Time: 40 mins

In C++, we can change the way operators work for user-defined types like objects and structures. This is known as operator overloading. For example, suppose we have created three objects c1, c2 and result from a class named Complex that represents complex numbers. Since operator overloading allows us to change how operators work, we can redefine how the + operator works and use it to add the complex numbers of c1 and c2.

In this question, you are required to overload the + and - operators of a class named "**BookShelf**" which represents **one** real bookshelf. You are also required to create a copy constructor in your class as a function.

BookShelf needs to have the following **private** variables:

1. **shelf** (type integer pointer, it points to an array which contains the identification numbers of the books on the real bookshelf)
2. **no_of_books** (type integer, it is the number of books on the real bookshelf)
3. **genre** (type string, all books on a **single** real bookshelf have the same genre so this variable contains just one string and not an array of strings)

BookShelf needs to have the following **public** functions:

- Constructor ***[2 marks]***
- Parametrized Constructor ***[5 marks]***
- Function that overloads + operator ***[10 marks]***
    - This function will be used to add 2 BookShelf objects to make a new BookShelf object. See sample output for more info.
- Function that overloads - operator ***[15 marks]***
    - This function will be used to subtract 2 BookShelf objects to make a new BookShelf object. Check sample output for more information.
- Copy constructor ***[8 marks]***
- Destructor ***[3 marks]***
- Getter and setters for all the three variables ***[5 marks]***
- Print() function that prints like the sample output. Make a main() function too. ***[2 marks]***

Values of member variables of BookShelf class **must not be hardcoded**. They must be taken as **input from the user**. The variable "**shelf**" points to an array of integers that holds the identification number of each book on the real bookshelf. The identification number of the book at the ith position in the real bookshelf is at the ith index of the variable "shelf" which, as mentioned earlier, is an array of integers. Number of books on the real bookshelf and genre are to be decided by the user.

Sample Output # 1 (Overloaded – operator at work)

```
//FIRST OBJECT:
No of Books: 5
Genre: Comedy
Identification numbers are as follows:
0
1
2
3
4
//SECOND OBJECT:
No of Books: 5
Genre: Horror
Identification numbers are as follows:
10
11
12
13
14
//THIRD OBJECT OBTAINED FROM: SECOND OBJECT-FIRST OBJECT:
No of Books: 0
Genre: None
Identification numbers are as follows:
[Finished in 0.5s]
```

```
//FIRST OBJECT:
No of Books: 3
Genre: Comedy
Identification numbers are as follows:
0
1
2
//SECOND OBJECT:
No of Books: 5
Genre: Horror
Identification numbers are as follows:
10
11
12
13
14
//THIRD OBJECT OBTAINED FROM: SECOND OBJECT-FIRST OBJECT:
No of Books: 2
Genre: Horror
Identification numbers are as follows:
10
11
[Finished in 0.5s]
```

```
//FIRST OBJECT:
No of Books: 7
Genre: Comedy
Identification numbers are as follows:
0
1
2
3
4
5
6
//SECOND OBJECT:
No of Books: 5
Genre: Horror
Identification numbers are as follows:
10
11
12
13
14
//THIRD OBJECT OBTAINED FROM: SECOND OBJECT-FIRST OBJECT:
No of Books: 0
Genre: None
Identification numbers are as follows:
[Finished in 0.5s]
```

Sample Output # 2 (Overloaded + operator at work)

```
//FIRST OBJECT:
No of Books: 7
Genre: Comedy
Identification numbers are as follows:
0
1
2
3
4
5
6
//SECOND OBJECT:
No of Books: 5
Genre: Horror
Identification numbers are as follows:
10
11
12
13
14
//THIRD OBJECT OBTAINED FROM: SECOND OBJECT+FIRST OBJECT:
No of Books: 12
Genre: Horror and Comedy
Identification numbers are as follows:
10
11
12
13
14
0
1
2
3
4
5
6
[Finished in 0.6s]
```

Best of Luck!