Lab 3: Passing/returning objects

Total marks: 30 marks (10+20)

Deadline: same day before 11:59 pm (sharp)

Submission instructions

1. Submit your file in a .zip or .rar format. Name your file with your Roll number (e.g. BSEF19M009.zip or BSEF19M009.rar)
2. Make a separate folder for each lab task (e.g. task1, task2…). All the headers and .cpp files must be in respective folder with a screenshot of the output of a program of the respective task.
3. Every Cpp file must contain your name and roll number(In comments) at the top of each program
4. Don't enclose your code in comments otherwise it will not be evaluated.

Instructions: (MUST READ)

- **No compensation or makeup lab.**
- **Don't discuss with peers. Changing variable names/ changing for to while loop will not help you in hiding cheating attempt!**
- **You are not allowed to ask TA to verify/ prove your cheating case! Any such complaint from TAs will result in serious consequences. Don't expect any positive response from TAs in such regard.**
- **Cheating cases will be given ZERO first time. Second attempt to cheat will result in deduction in sessionals. I'll report the regular cheating attempts to Discipline committee without any prior warning.**
- **You are not allowed to consult Internet. Plagiarism cases will be strictly dealt.**
- **You can ask relevant queries (in case you are stuck somewhere or a very VALID query regarding some error) from TAs only between 9 am – 12pm slot.**
- **You are not allowed to ask TAs about problem statement (e.g. what to write in constructor/how many classes should be made/what to write in header file etc..These are all INVALID queries). Understanding the problem statement is a part of that question. Do whatever you are asked and what you understand.**
- **You must submit the lab solution BEFORE 11:59 pm. Even a few minutes late submissions will not be considered. Make sure to do proper management of time/Internet connectivity/power failure or whatever issue is possible!**

General instructions for all tasks: (marks will be deducted if the instructions are violated)

Note: All the programs should be implemented using class. You can take input in main() function and then call appropriate methods/ member functions of a designed class to set and get values. YOU MUST CREATE A SEPARATE CPP AND HEADER FILE(S) FOR CLASS DECLARATION AND DEFINITION.

• The attributes of class should be declared as **private** and member functions as **public**.

• All the member functions (expect constructor) should be declared inside the class and defined outside the class.

• You should not initialize the attributes while declaring them in class. The values should be assigned using member functions only. E.g. you cannot declare like:

```
Class Person
{
        Private:
        int age=25;
}
```

• The values should be initialized using a constructor. There must be a constructor in your defined class.

• All inputs should be taken in *main()* and all the final results should also be reported/ displayed in the main function.

• All the logic should be implemented in class' member functions. Main() should only input and output relevant values by calling relevant functions of the class.

Question 1: **10 marks**

Create a class **ArmyOfficers** that has data members as *id,* and *rank*. The class should have following methods:

Default constructor that initializes data members to 0.

Overloaded constructor that sets the values passed as an argument to each of the data members.

Void set(int id, int rank): Set method that takes name, id, and rank as an argument and set these values to relevant data members.

Int getID(), int getRank(): Get method for each of the data members that does not take any argument but returns the value of a data member.

ArmyOfficers getHighestRank(ArmyOfficers a): Get method that takes an object of class as an argument. The function then compares the rank of **object a** with current objects' rank. The function then returns the object that has highest rank.

The main function does the following:

• The main function should declare two objects. It should declare one of the objects using overloaded constructor(hardcode the data members using values of your choice). It should also display the hardcoded values by calling the get methods of class.

• It should then take input from the user (id and rank of the objects other than the one declared using overloaded constructor) and update the data members of that object.

(Void set(int id, int rank))

• It should then call *getHighestRank* function by passing one of the objects as an argument. The function should then return the object that has higher value of rank attribute. Main() should then display the id and rank of that object.

ArmyOfficers getHighestRank(ArmyOfficers a)

Expected input/ output:

Enter the ID of army officer: 11

Enter the rank: 5

ID of army officer (hardcoded): 10

Rank (hardcoded): 4

ID of highest ranked officer: 11

Rank: 5

Question 2: **20 marks**

Create a class **Bag20** that contains a collection of 20 numbers. Each bag is initially empty. This is called the initial state of a bag. (Think of a bag20 as an statically defined array of 20 integers). The class will contain two attributes: integer array of size 20 and current size (number of elements in array, i.e. 0 initially).

*You need to implement following **operations/ funtions** on bag20:*

***Default constructor** for initializing array and size variable*

Void Insert (int);

Insert a number in the bag. This function will receive user entered number from main() and insert that into data member (array).

Void show();

The function should display the contents of an array

Int elementAt(int);

Query a bag contents. This function will take an index as an input from the user and return the value present at that index. You need to perform validation if the user entered index is greater (goes out of bound) that the current size of bag. For example, if the bag20 array has 3 elements (1,55,3) and user asks to tell the value at index 6, then this function should output error message “index goes out of bound” and return -1. In other case, the value of array present at that index number is returned.

Int findFrequency(int);

This function will find how many copies does the bag contains of a given number (received as argument). For example, if array has elements (3,1,2,3,3,55,3,1) then if '3' is passed to this function then it should return 4 (the number of times 3 exists in an array).

Void delete (int);

Remove a single number from the bag. This will remove the entered number from an array. The function will first find if the number (passed to it as an argument) exists in an array. If so, then the function should delete all the occurrences of this number. The size of an array should also be decremented. For example, if array has elements (3,1,2,3,3,55,3,1) then if '3' is passed to this function then final array should be (1,2,55,1).

Bag20 union (Bag20);

Add one bag to another. This function should accept an object as an argument. The function should then merge/ take union of two bags/arrays and return the merged array

Int findEquilibriumIndex(Bag20);

This function returns an equilibrium index of a bag. An index of an array is said to be equilibrium index if the sum of the elements on its lower indexes is equal to the sum of elements at higher indexes. For Example:

A[] = {7, -1, -5, 2, 4, -3, 0} Output : 3 is an equilibrium index, because: A[0] + A[1] + A[2] = A[4] + A[5] + A[6] = 1

If no equilibrium index is found then your function should return -1.

Bag20 leftRotate(Bag20,int);

This function should rotate the bag 3 (received as an argument) by the factor (received as a second argument). For example, rotating {1,2,3,4,5,6,7} by factor 2 will result into bag: {3,4,5,6,7,1,2}. After rotation, create a temporary bag (e.g. bag 4) with size as (size of bag 1+size of bag 3). Place the entries of resultant bag at odd indexes of bag 4 and contents of bag 1 are at even index of bag (i.e. 0,2,4…) If size of bag 1 is less than bag 3's size then you can place 0 at extra cells.

The main function does the following:

- The main function should declare two objects.
- It should then add take input from the user (elements) and update the contents of arrays of two objects', respectively. The program should keep on prompting input from the user till two conditions are satisfied: 1) count of input values reach 20, 2) user enters -1. *(**insert** function should be invoked)*
- The user should display the contents of both bags' contents (arrays). *(**show** function should be invoked)*
- The user should be asked about the index at which value should be found in bag 1 array. *(**elementAt** function should be invoked)*
- The user should be asked about the element whose frequency should be found in bag 1 array. *(**findFrequency** function should be invoked)*

- The user should be asked about the element which should be removed from bag 1 array. *(**delete** function should be invoked).* ***Show*** function should be invoked right after this function to display the contents of bag 1.
- Invoke the function union and pass bag 2 to it. Take the result in another object bag 3 *(**union** function should be invoked)*
- Invoke the function union and pass bag 3 to it. *(**findEquilibriumIndex** function should be invoked)*
- The user should be asked about the factor by which to rotate bag 3 array. *(**leftRotate** function should be invoked).* Create bag 4 object and receive the object returned by this function. ***Show*** function should be invoked right after this function to display the contents of bag 1 and bag 3 and bag 4(returned by function).

Expected input/ output:

****Enter the elements for two bags****

Enter the value to insert in bag 1: 2

Enter the value to insert in bag 1: 33

Enter the value to insert in bag 1: -1

Enter the value to insert in bag 2: 22

Enter the value to insert in bag 2: 5

Enter the value to insert in bag 2: 6

Enter the value to insert in bag 2: -1

Contents of bag 1: 2 33

Contents of bag 2: 22 5 6

Enter the index at which value should be found in bag 1 array: 1

Value at index 1: 33

Enter the element whose frequency should be found in bag 1 array: 33

Frequency of 33: 1

Enter the element which should be removed from bag 1 array: 33

Contents of bag 1: 2

Union of bag 1 and bag 2 (bag 3):

2 22 5 6

Equilibrium index of bag 3: -1

Enter the factor by which to rotate bag 3: 1

Contents of bag 1: 33

Contents of bag 3: 22 5 6 2

Contents of bag 4: 33 22 0 5 0 6 0 2