Lab 9: Binary files and virtual functions

Total marks: 30 marks (15+15)

Deadline: same day (Friday) 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: **15 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.

The main function does the following:

1. The main function should declare array of 10 objects. It should then take input from the user and update the data members of each object. Side by side it should also write each object to a binary file (ArmyOfficers.dat).
2. Implement a function named as findHighRanks(). This function should open a file (ArmyOfficers.dat) and read the objects one by one. While reading the objects it should sort the objects with respect to their ranks and prepare an integer statically defined array named as ***Ranking***. This array should be declared as global so as to be accessible to the whole program. The cell of this array contains the sequence number of armyOfficer with respect to rank. i.e. if we have ranks as 2,11,5,27,33,10,6,7,8,13 then the order array Ranking would enclose following entries:
   10, 4,9,2,1,5,8,7,6,3
   First entry is 10 because the rank of first army officer (i.e. 2) is the least among the others. Similarly, rank of second army officer (i.e.11) is 4 which means fourth highest.

Your function or logic should be generic and should involve loop! Not a bunch of if else statements otherwise zero will be granted. For similar ranks, you can handle this case accordingly i.e. if two ranks are same then either give high rank to the first entry or the second entry.

Marks distribution:

Class construction: 2 marks

Main function (1): 5 marks

Main function (2): 8 marks

Question 2: **15 marks**

Radio Frequency IDentification (RFID) chips are small tags that can be placed on a product. They behave like wireless barcodes and can wirelessly broadcast an identification number to a receiver. One application of RFID chips is to use them to aid in the logistics of shipping freight. Consider a shipping container full of items. Without RFID chips, a human has to manually inventory all of the items in the container to verify the contents. With an RFID chip attached to the shipping container, the RFID chip can electronically broadcast to a human the exact contents of the shipping container without human intervention.

To model this application, write a base class called ShippingContainer that has a container ID number as an integer. Include member functions to set and access the ID number. Add a virtual function called getManifest that returns an empty string. The purpose of this function is to return the contents of the shipping container.

Create a derived class called ManualShippingContainer that represents the manual method of inventorying the container. In this method, a human simply attaches a textual description of all contents of the container. For example, the description might be "4 crates of apples. 10 crates of pears." Add a new class variable of type string to store the manifest. Add a function called setManifest that sets this string. Override the getManifest function so that it returns this string.

Create a second derived class called RFIDShippingContainer that represents the RFID method of inventorying the container. To simulate what the RFID chips would compute, create an add function to simulate adding an item to the container. The class should store a list of all added items (as a string) and their quantity using the data structures of your choice. For example, if the add function were invoked three times as follows:

```
rfidContainer.add("crate of pears"); // Add one crate of pears
rfidContainer.add("crate of apples"); // Add one crate of apples
rfidContainer.add("crate of pears"); // Add one crate of pears
```

At this point, the data structure should be storing a list of two items: crate of apples and crate of pears. The quantity of apples is 1and the quantity of pears is 2. Override the getManifest function so that it

returns a string of all items that is built by traversing the list of items. In the example above, the return string would be “2 crate of pears. 1 crate of apples.”

Finally, write a main program that creates an array of pointers to six ShippingContainer objects. Instantiate the array with three ManualShippingContainer objects and three RFIDShippingContainer objects. For the ManualShippingContainer objects, you will have to invoke setManifest to set the contents. For the RFIDShippingContainer objects, you will have to invoke add to set the contents (although, if this were real, the contents of the container would “add” themselves via the RFID chips instead of requiring a human to type them in). Finally, write a loop that iterates through all ShippingContainer pointers and outputs each object’s manifest along with the shipping container ID. You may need to convert an integer into a string. A simple way to do this is: string str= to_string(intVar)