Lab 4: Pass by reference, this pointer

Total marks: 20 marks (10+10)

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.**
- **Understanding the problem statement is a part of that question. Do whatever you are asked and what you understand.NO QUERIES ARE ALLOWED as this lab is overly simple**
- **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.

Task 1

You are required to implement a class of **Element,** with five attributes. Each element has

1. ID (integer),
2. size (static integer attribute) that consists of number of elements in the system,
3. a pointer (integer pointer) that stores address of the memory location that this elements points to.
4. Alpha (static integer attribute)
5. Beta (static integer attribute)

Size, alpha, and beta should be initialized with zero. For example, following elements are shown with their memory addresses shown at the bottom.

Memory location 1	Memory location 2	Memory location 3
ID: 11Pointer: 0XB000 Size:3 Alpha: 1 Beta: 0	ID: 12 Pointer: 0XB001 Size:3 Alpha: 0 Beta: 1	ID: 13 Pointer: 0XB000 Size:3 Alpha: 1 Beta: 0
0XA000	0XA001	0XA002

The class will have following functions:

Element() Default constructor that initializes ID with zero and pointer with null. Moreover, it increments the size attribute by 1.

Void setVal(int ID, int alpha, int beta) that initializes ID, alpha, and beta with the value received in arguments using **this** pointer (must use this).

Void setPointer() that initializes pointer with either alpha's address or beta's address using **this** pointer (must use *this*). If the alpha value of an object (represented by *this*) is 1, then the pointer should be made pointed to alpha. Otherwise, it should point to beta (initialized with beta's address).

Static int getSize () that returns size attribute.

int getID () that returns ID attribute.

int getAlpha () that returns Alpha attribute.

int getBeta () that returns Beta attribute.

Void printPointerAddress(Element e) that prints the contents of pointer attribute of the object passed in arguments.

You have to perform following steps:

1) In main(), ask the user about the number of elements that are required to be kept. Declare an array of **elements** according to the entered size.
2) Take ID (integer value) of each **element** from the user in main(). Loop through the array of objects (created in step 1) and set the user entered ID value to each object's class member using *setID* method. You must use *this* pointer inside *setID* method. If the index of object's array is even then alpha should be updated with 1 and beta with 0. Otherwise, alpha should be updated with 0 and beta with 1.
3) Set the pointers using *setPointer* method. You might need to iterate through an object array to set the pointer value of each object.
4) Print the addresses of alpha and beta along with the following for each element. Use getter and print methods defined.
    a. ID
    b. Pointer's address
    c. Size

Sample run:

Enter the number of elements: 3

Enter the ID for element 1: 11

Enter the ID for element 2: 12

Enter the ID for element 3: 13

Alpha: 0XB000

Beta: 0XB001

Element# 1

ID: 11

Pointer address: 0XB000

Size:3

Element# 2

ID: 12

Pointer address: 0XB001

Size:3

Element# 3

ID: 13

Pointer address: 0XB000

Size:3

Task 2

```
class Rational
{
        int p,q;
        public:
        Rational(int, int); //call set
        void set(int , int); //check q if equal to 0 then set 1, otherwise pass value to q
        void show() const;
        void simplify();//divide p & q by common divisor (if any), repeat till p & q has no common divisor
        Rational add(Rational &r);//add 2 rational numbers and return result in 3rd rational number
        Rational subtract(Rational &r);//Subtract and return result in 3rd rational number
        Rational mul(Ration &r);//multiply 2 rational numbers and return result in 3rd rational number
};
```

Write main function separately to test functions of your class.