IOOM - Object Oriented Programming

Lab Assignment – 3

Submission Date & Time: Friday 26, March, 2021, 11.00 PM

Note: i) Input should be taken only from the user and not hard coded.
ii) You need to make all data members either private or default.
iii) Make all methods either public or default.

Batch R1 & R2

Problem 1:
Create a class 'item' that contains the following data fields:
1. item_id(integer)
2. Quantity(integer)
3. item Label(character array of size 10)
and a suitable constructor .

Create another class 'Stack' which contains an array of such items and data member 'top' and 'size'. In the constructor, initialize the top to -1.

Create methods to push( ), pop( ) , is_empty( ) and display( ) the stack. Note that the push method will call the constructor of the 'item' class.

Problem 2:
Augment the above program to include the following two methods in the stack class:

a. A method to sort the elements in the stack only using the class 'Stack' methods defined previously (push(),pop(), is_empty()) such that the item with the largest quantity appears at the top.

b. A method that removes the middle element in the stack (using only the standard stack methods defined in question 1).

For both questions, create a main method. Create an object for Stack and verify the correctness of the methods you've defined by invoking them appropriately.

Problem 3:
Write an object oriented program for manipulating strings by creating your own definitions of operators +, <=, = =, **!=** and subscript operator [ ] using friend functions as follows.

i) The declaration for the String class looks as follows:

```
    Class String{
          char *value;
            int length;
          public:
            Methods:-
           Constructors:
           string ( ) { length=0; value=0);
           string (const char *s) ;
           string ((const string &s);
           //operators
           friend string operator + (const string &s, const string &t);
           friend int operator >=  (const string &s, const string &t);
           friend int operator== (const string &s, const string &t);
           friend int operator !=  (const string &s, const string &t);
           friend char operator [ ] (int value);   //(Unary)
           friend void display(const string s);
};
```

ii) Write definition of each function and use given operators for manipulating strings. Take different user inputs for string.

Problem 4: Define a custom class LinkedList with appropriate constructors, destructors, data members and functions which contain ComplexInteger objects as elements. Also 2 additional friend functions to display the contents of the LinkedList and Search for the Specified object and returning boolean value (if present or not). Also perform the following operations via its Objects by overloading appropriate operators.

i) + : Adds a ComplexInteger object at the end of the list.(when called with a ComplexInteger Object).

And

Appends another list at the end (when called and passed another List Object).

ii) - : Deletes a ComplexInteger object from the end of the list.

iii) * : Deletes the element at the specified no. from the beginning. (if it exists)

```
        Say list is :
                2+3i → 5+2i → 7+9i
        After calling :
                list*2;
        The list becomes :
                2+3i → 7+9i
                (element no.2 deleted )
```

iv) = : Assigns a list Object to another list reference (when called with 2 LinkedList objects).

And

Creates a fresh List from an array of ComplexInteger Objects. (when called from a list object and passed an array of ComplexInteger objects).

v) << : Left Shifts the contents of the list specified no. of times.

Ex: list<<2;

vi) >> : Right Shifts the contents of the list specified no. of times.

Ex: list>>3;

Batch R3 & R4

Problem 1:

1) Design a class to evaluate a given postfix expression using stack and operator overloading. The operands are complex numbers of user defined type "Complex". Your class definition should contain the class members : real and imaginary of type int. The class definition should also contain the setComplex() function that sets the real and imaginary part of the operand and display() function that displays the result. The stack should be an array of type Complex.

   Write member functions for following binary operators:

   " +, -, *, /".

   Also overload << and >> operator.

   Sample input : (5 3) (2 1) (1 5) * + (1 4) - [which implies (5+3i) (2+i) (1+5i) * + (1+4i) -]

   Output: -1 -11i

   Explanation: Evaluating from left

   Push (5 3) on to the stack since it is an operand

   Push (2 1) on to the stack since it is an operand

   Push (1 5) on to the stack since it is an operand

   Since we get an operator (*) next, pop two element from the top of the stack and perform the operation.

   (1 5) * (2 1) [i.e., (1+5i)*(2+i) = (-3+11i)]

   Push the result (-3 11) on to the stack

   Continue until the end of the string is reached.

Problem 2: Define a custom class ComplexInteger with appropriate data members, constructors, copy constructor, destructor, getter methods setter method and functions. Also an additional friend functions to display the contents of the Object. Perform the following arithmetic operations via its Objects by overloading appropriate operators.

i) + : Adds 2 ComplexInteger objects.
ii) - : Subtracts a ComplexInteger object from the other.
iii) * : Multiplies 2 ComplexInteger objects.
iv) / : Divides 2 ComplexInteger objects.
v) = : Assigns a complex Integer or a Real Integer to a ComplexInteger object.
vi) <> : (Unary) Swaps the real and Imaginary values of the object.
vii) ^ : (Unary) Multiplies the ComplexInteger object with its Conjugate and stores in the current object.

Problem 3: Define a class Matrix with appropriate constructors, copy constructor, destructor, getter methods setter method and functions which contain ComplexInteger objects as an elements. Also 2 additional friend functions to display the contents of the Matrix and Search for the Specified object and return boolean value (if present or not).

Also perform the following operations via its Objects by overloading appropriate operators.

i) + : Adds a scalar (ComplexInteger Object) to every element of the Matrix (when called with a ComplexInteger Object or a Real Integer).

And

Adds respective members of the Matrix (Only applicable for matrices of same order as and when called with another Matrix Object).

ii) - : Subtracts a scalar (ComplexInteger Object) from every element of the Matrix (when called with a ComplexInteger Object or a Real Integer).

And

Subtracts respective members of the second Matrix from the first one. (when called with another Matrix Object).

iii) ^ : Replaces the element at the specified no. from the beginning in natural counting order with 0+0i. (if it exists)

```
Say Matrix is :
        2+3i    5+2i    7+9i
        5+3i    10+2i   71+9i
        8+3i    15+2i   27+19i

After calling :
        matrix^7;
The Matrix becomes :
        2+3i    5+2i    7+9i
        5+3i    10+2i   71+9i
        0+0i    15+2i   27+19i
        (element no.7 replaced with 0+0i )
```

iv) = : Assigns a Matrix Object to another Matrix reference (when called with 2 Matrix objects).

And

Creates a fresh List from a 2D array of ComplexInteger Objects. (when called from a matrix object and passed a 2D array of objects).

v) * : Multiplies a scalar (ComplexInteger Object) to every element of the Matrix (when called with a ComplexInteger Object or a Real Integer).

And

Performs Matrix Multiplication for 2 matrices. (when called together with another Matrix Object).
