CL-217
Object Oriented Programming
Lab # 13

Objectives:

- Operator Overloading

Note: Carefully read the following instructions (*Each instruction contains a weightage*)

1. There must be a block of comments at start of every question's code by students; the block should contain brief description about functionality of code.
2. Comment on every function about its functionality.
3. Use understandable name of variables.
4. Proper indentation of code is essential
5. Write a C++ statement(s) for each of the following task one after the other, in the same order.
6. Make a Microsoft Word file and paste all of your C++ code with all possible screenshots of every **task outputs in MS word and do not submit .cpp file with word file**.
7. First think about statement problems and then write/draw your logic on copy.
8. After copy pencil work, code the problem statement on MS Studio C++ compiler.
9. At the end when you done your tasks, attached C++ created files in MS word file and make your submission on Microsoft teams. (Make sure your submission is completed).
10. Please submit your file in this format 19F1234_L9.
11. Do not submit your assignment after deadline.
12. Do not copy code from any source otherwise you will be penalized with negative marks.

Problem 1: Operator Overloading

Write a class Employee having following attributes:

1. String name
2. Integer Age
3. Float Salary

Over load the appropriate operators for the following functionality:

1. Adding two employee objects (concatenating the names of both objects, add the rest of the two data members)
2. Telling which employee is elder (overload – operator for this)
3. Comparing the salary of two employees
4. Input employee object

5. Output employee object

Write another class Person having the following attributes:

1. String name
2. Integer Age

Convert an object of type Employee to person. You may use this pointer for this purpose.

Problem 2: Operator Overloading

Write a class **Factorial** to overload ! operator to find factorial of an integer object. Write a driver program to test your class. Show input and output results on console.

Problem 3: Operator Overloading

Define a class **Matrix** to represent rows × cols matrix. r (row) and c (column) will be passed as parameters to constructor of class Matrix:

1. Overload operators for addition (use "+" operator for addition) and subtraction (use "-" operator for subtraction) of two matrices.
2. Overload operators for pre-increment (use "++" operator. This operator will increment (all elements of matrix by) 1.
3. Overload operators for post-increment (use "++" operator. This operator will increment (all elements of matrix by) 1.
4. Overload operators for pre-decrement (use "--" operator. This operator will decrement (all elements of matrix by) 1.
5. Overload operators for post-decrement (use "--" operator. This operator will decrement

(all elements of matrix by) 1.

6. Overload insertion ">>" to input all elements of matrix.
7. Overload extraction "<<" operator to output all elements on console.
8. Overload less than operator "<" for two matrices. This operator will return true if the sum of all elements of matrix is less than second. i.e. A < B.
9. Overload less than operator ">=" for two matrices. This operator will return true if all elements of matrix A is greater than or equal to second. i.e. A >= B. If any one element is smaller than B at same location, it will return false.
10. Overload unary operators "*" that will return the product of all elements of a matrix.

Note: Size of matrix A will be same as size of B for binary operators.

Write a driver program to test your class.

Problem 4: Operator Overloading

Write a class Complex for complex numbers having the following data members:

1. Float a
2. Float b

Write overloaded and default constructors for your class.

Implement the following functionality for your class.

float mag( );	It will compute and return the magnitude of a complex number. The magnitude of a complex number $a + bi$ is the quantity $\sqrt{a^2 + b^2}$.
Complex add(Complex c);	The method accepts a complex number c, adds it with *this* complex number and returns the answer as another complex number. The addition of two complex numbers, $a + bi$ and $c + di$ is defined as follows: $(a + bi) + (c + di) = (a + c) + (b + d)i$
Complex mul(Complex c);	The method accepts a complex number c, multiplies it with *this* complex number and returns the answer as another complex number. The multiplication of two complex numbers, $a + bi$ and $c + di$ is defined as follows: $(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$
Complex operator++;	Overload pre-increment operator
Complex operator++(int);	Overload post-increment operator
Complex operator--;	Overload pre-decrement operator
Complex operator--(int);	Overload post-decrement operator
ostream& operator<<(ostream& os, const Complex& c);	Overload extraction operator so that it can display *this* complex number, in the format: **a+bi**