CL-1004
Object Oriented Programming
Lab No 11

Objectives:

- Operator Overloading
- Friend function and friend class

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 output in MS word and submit .cpp file with word file**.
7. Make separate .cpp files for all tasks and use this format **22F-1234_Task1.cpp.**
8. First think about statement problems and then write/draw your logic on copy.
9. After copy pencil work, code the problem statement on MS Studio C++ compiler.
10. At the end when you done your tasks, attached C++ created files in MS word file and make your submission on Google classroom. (Make sure your submission is completed).
11. Please submit your file in this format **22F-1234_L1**.
12. Do not submit your assignment **after deadline**.
13. **Do not copy code from any source otherwise you will be penalized with negative marks.**

Problem 1: Friend class

You are working on a project to build a text-based adventure game in C++. The game has different game characters, such as warriors, mages, and archers, who have unique abilities and attributes. You are given three classes - **Character, Warrior, and Mage** - that are part of the game.

The Character class represents a generic game character and has the following attributes:

name: a string representing the name of the character.

level: an integer representing the level of the character.

health: an integer representing the health points of the character.

The Warrior class represents a warrior character. It has the following additional attributes:

weapon: a string representing the weapon used by the warrior.

strength: an integer representing the strength of the warrior.

The Mage class represents a mage character. It has the following additional attributes:

spell: a string representing the spell used by the mage.

intelligence: an integer representing the intelligence of the mage.

You need to implement the following functionalities:

Create a new character with a given name, level, and health.

Create a new warrior character with a given name, level, health, weapon, and strength.

Create a new mage character with a given name, level, health, spell, and intelligence.

Display the details of a character, including its name, level, health, and additional attributes (weapon and strength for warriors, spell and intelligence for mages).

Update the health points of a character.

Use the abilities of a warrior character, such as attacking with its weapon and using its strength.

Use the abilities of a mage character, such as casting a spell and using its intelligence.

You should use appropriate inheritance, member functions, and friend classes in C++ to implement the above functionalities. Ensure that the classes have appropriate access modifiers and that the appropriate data encapsulation principles are followed.

Your task is to implement the Character, Warrior, and Mage classes with appropriate member functions and friend classes, and demonstrate their usage in a C++ program that simulates the text-based adventure game.

Problem 2: Friend function

Implement a class hierarchy for a multimedia library that includes different types of multimedia items, such as books, videos, and music albums. Each multimedia item should have common features such as a title, creator, and publication year. The MultimediaItem class should be the base class for all multimedia items, and it should have private data members for title, creator, and publication year, along with appropriate member functions to set and get these values.

The Book class should be derived from the MultimediaItem class and should add additional features such as author, genre, and number of pages. The Video class should also be derived from the MultimediaItem class and should add features such as duration, format, and resolution. The MusicAlbum class should also be derived from the MultimediaItem class and should add features such as artist, genre, and number of tracks.

Implement a friend function called displayDetails() that takes an object of the MultimediaItem class as an argument and displays the details of the multimedia item, including the common features (title, creator, publication year) as well as the additional features specific to each derived class (author, genre, number of pages, duration, format, resolution, artist, genre, number of tracks).

Using friend functions, you can access the private members of the MultimediaItem class from the derived classes (Book, Video, MusicAlbum) to display their details, even though the friend function is not a member of the class hierarchy. This allows for proper encapsulation while providing access to private members for specific purposes.

Problem 3: 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 4: Operator Overloading

Write a class Employee having following attributes:

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

Overload the appropriate operators for the following functionality:

1. Input employee object(cin>>obj)
2. Adding two employee objects (concatenating the names of both objects, add the rest of the two data members) (obj3 = obj1+obj2)
3. Telling which employee is elder (overload – operator for this)
4. Comparing the salary of two employees
5. Output employee object(cout<<obj)

Use 3 file structure

Problem 5: 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 first matrix is less than second. i.e. A < B.
9. Overload less than equals to 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.

Use 3 file structure

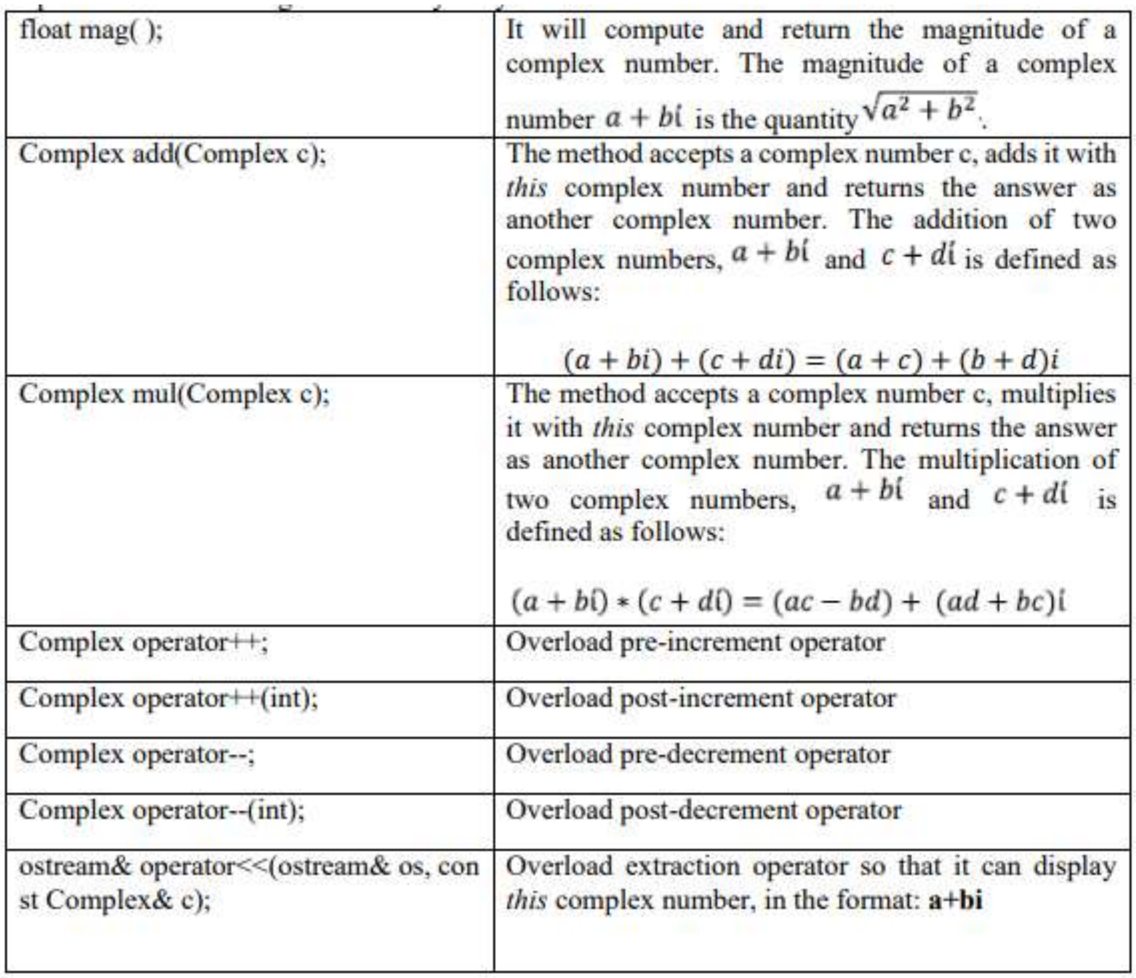
Problem 6: Operator Overloading

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**

Use 3 file structure

Proper code indentation will hold extra marks !

Best of luck ☺

You are done with your exercise, submit on classroom at given time.