CL-1004
Object Oriented Programming
Lab No 11

Objectives:

- Composition
- Aggregation
- 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 and about its functionality.
3. Mention comments where necessary such as comments with variables, loop, classes etc to increase code understandability.
4. Use understandable name of variables.
5. Proper indentation of code is essential.
6. Write a code in C++ language.
7. Make a Microsoft Word file and paste all of your C++ code with all possible screenshots of every task **output in Microsoft Word and submit word file.**
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 **20F1234_L1**.
12. Do not submit your assignment after deadline. Late submission is not accepted.
13. Do not copy code from any source otherwise you will be penalized with negative marks.
14. Submit .cpp files of all tasks.

Problem 1: Composition

Create a class Time with following data members and member functions

```
public:
	Time();
	Time(int, int);
	void setTime(int, int);
	void getTime(int&, int&);
	void printTime();
private:
	int hr;
	int min;
```

Create a class Date with following data members and member functions

```
public:
	Date();
	Date(int, int, int);
	void setDate(int, int, int);
	void getDate(int&, int&, int&);
	void printDate();

private:
	int month;
	int day;
	int year;
```

Create a class Event with following data members and member functions

```
public:
	Event(int hours = 0, int minutes = 0, int m = 1,
		int d = 1, int y = 1900, string name = "Christmas");
	void setEventData(int hours, int minutes, int m, int d, int y, string
name);
	void printEventData();

private:
	string eventName;
	Time eventTime;
	Date eventDay;
```

Write a Main and create events using the above classes to demonstrate composition.

Problem 2: Death Relation

Implement the death relation using car as a whole class and engine, wheel, window, door and tires as a part classes.

Problem 3: Aggregation

Implement the person address problem with respect to aggregation. You need to make your code in such a way that 3 persons living on a same address

Problem 4: Aggregation

Implement the person teacher problem with respect to aggregation. You need to make your code in such a way that if teacher object is destroyed still person exists.

Problem 5:

Mention at-least five examples of composition and five examples of aggregation with proper explanation.

Problem 6: 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 7: 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 8: 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.