Week 6:Operator Overloading, Friend Function

Learning Materials: Chapter 8

1. **[You can use your solution code of previous lab task, Read the following instruction VERY CAREFULLY as you must refactor the code for today's task]**
2. **[Make the member function const and make the parameter const where it is possible.]**
3. **[While passing an object, use the Reference variable in the parameter, Do not use the pass by value method.]**

Task 1

Create a class called "Matrix3D". An object of the Matrix3D class stores a three dimensional matrix.
Implement the following public member functions (the task of the function is written after a hyphen):

- Necessary constructor and destructor functions.
- double det() : Returns the determinant of the matrix.
- Matrix3D inverse(): returns the inverse matrix.

Overload binary operator +, -, * according to the matrix operations.
Your code should support following code:

```
mat1 = mat2 + mat3;
mat1 = mat2 - mat3;
mat1 = mat2 * mat3;
```

Task 2

Create a class "**Coordinate**". An object of the **Coordinate** class stores the abscissa and ordinate (float type).
Implement the following **public** member functions (task of the function is written after a hyphen):

- **Necessary constructor, destructor and display function.**
- **float getDistance(Coordinate c)** - Distance from object c
- **float getDistance()** - Distance from origin (0,0) coordinate

- **void move_x(float val)** - val will be added to member data abscissa
- **void move_y(float val)** - val will be added to member data ordinate
- **void move(float val) -** move_x(val) and move_y(val) will be called

Write the necessary member or non member functions to achieve the following functionalities.

- Assume c1,c2,c3 are Coordinate objects. Overload the following comparison operators >,<,>=,<=,==,!= where distance from the origin of each operand will be compared. Example c1 == c2 returns true when c1 contains (abscissa = 1, ordinate = 1) and c2 contains (abscissa = -1, ordinate = -1)
- Unary operator ++ will move a coordinate object 1 unit in x and y directions. Implement prefix and postfix according to the convention.
- Unary operator -- will move a coordinate object -1 unit in x and y direction. Implement prefix and postfix according to the convention.

Task 3

Create a class **FLOAT** that contains **one float data member.** Suppose f1,f2,f3,f4 are four objects of class **FLOAT**.Overload all the four arithmetic operators(+,-,*,/) so that they can operate on the objects of **FLOAT**.For example: ***FLOAT operator+(FLOAT f1)*** for (+) operator.

- Also implement an operator overloading function to achieve the following functionalities.
  Float f1(5.052);
  int i = f1; // this will store value 5 in i

Task 4

Create a class named "**Counter**". An object of the Counter class keeps track of the count. The object also stores the value of the increment steps. For example, if the increment step is 5, the count will be incremented by 5 each time an increment is done. To do these, declare the **necessary private member** data.

Note that there should not be any public methods to assign a value to count.

Implement the following public member functions (task of the function is written after a hyphen):

- void setIncrementStep(int step_val) - it sets the value of Increment Step in the appropriate member data. The default value is 1. Restrict assigning a negative value. Keep the previous assigned value if a negative value is passed as an argument. However, the constructor assigns a default value if a negative value is passed.
- int getCount() - it returns the current count value.
- void increment() - it increments the count by increment step. For example: if the current count is 4 and the increment step value is 2 then executing the function will update the count to 6.
- void resetCount() - it resets the value of count to 0.

Write the necessary member or non-member functions to achieve the following functionalities.

- Assume c1, c2, c3 are Count objects. c1 = c2 + c3; After this statement, the count value of c1 will be the summation of the count of c2 and c3. No changes to c2 and c3. The step value will be the maximum of c2's and c3's.
- Implement >, <, >=, <=, ==, and != operator where they only compare the count value of the object and they follow their usual meaning. Example: c1==c2 will evaluate to true if the value of count variable of c1 and c2 are equal. Hints: Use bool variables.
- c1+=c2; No changes to c2. After this statement, the count value of c1 will be the summation of the count of c1 and c2. No changes to c2. The increment step value will be the maximum of c1's and c2's.

- c1 = c2++; c1= ++c2; functions similar to increment() function.

The return policy should follow conventional rules.

```
void testFunction(const Counter& c)
{
cout<< c.getCount();
}
```