Classes. Objects. Static members.Functions and classes friend.

Objectives:
-static members, examples
-static member functions
-functions friend
-classes friend

Static class members – study the examples 3.1 and 3.2 from lab 4

Static member functions can access only static members of a class, do not have the pointer this.
Example:

```
class Operations
{
static int a;
int b;
public:
      void set(int i,int j)
      {
      a=i;
      b=j;
      }
      static void print();

/* int print () //error cannot overload static and non-static
                  //member functions with the same parameter types
      {
      return a+b;
      }  */
      int print(int i,int j)
      {          //we can overload the function if we have
                  // a different numbers of parameters
      return i+j;
      }
};


int Operations::a;  //Defines the global variable
void Operations::print()
{
```

```
cout<<"Static member "<<a<<endl;
// cout<<b; //error illegal reference to non-static member
}

int main()
{
 Operations ob1,ob2;
 ob1.set (10,10);
 ob2.set (20,20);
 Operations::print();
 ob2.print();
 ob1.print();
 cout<<ob1.print(3,3);
}
```

Friend functions are functions which are not members of the same class for which they are friend and they can access the private members from the class for which they are friend. The functions can be members of other class or can be global functions.

Syntax**:** friend function_type function_name(list_of_parameters);

Example:

```
class Problem
{
int a,b;
public:
      friend int sum(Problem x);
      explicit Problem(int i,int j);
};

Problem::Problem(int i,int j)
{
a=i;
b=j;
}

int sum(Problem object)
{
//here we can access the private membersfrom the class Problem
      //because this function is declared as friend
return object.a+object.b;
}
```

```
int main()
{
Problem example(3,4);
cout<<"add 3 with 4 is "<<sum(example);
}
```

Friend classes A class Y is friend with a class X if all the members from the class Y are friend functions of the class X. (namely, all the members of the class Y can access all the members from the class X).

Syntax: friend class X;

Example:

```
class X;

class Y
{
    int a, b;
  public:
        Y(int i,int j)
        {
        a=i;
        b=j;
        }
    int f1()
      {
            cout<<"Message: ";
            return a*b;
        }
     void f2(); //has to be defined
     void convert (X x);
};


class X
{
    int c;
  public:
    void set (int f)
      {
            c=f;
        }
```

```
    friend class Y;
};

void Y::convert (X x) {
  a = x.c;
  b = x.c;
}

int main () {
  X obj1;
  Y obj2(2,5);
  cout<<obj2.f1()<<endl;
  obj1.set(4);
  obj2.convert(obj1);
  cout << obj2.f1();
  return 0;
}
```

Problems:

1. **Define and implement the class Rectangle,** which has the following members:

a) Alternative 1: Length and Width and the member functions: setLength, setWidth, getLength, getWidth, Area and Perimeter;

b) Alternative 2: Length and Width, a default constructor, and explicit constructor, a copy constructor, a destructor, a method which returns the **Area** of a rectangle, a method which calculates the **Perimeter** of a rectangle. Use the pointer *this*.

c) Exemplify for at least 3 objects.

d) Implement a **friend class Square** and calculate the perimeter of the square and the main diagonal of the square by considering the side of the square to be the Length of the rectangle**.**

e) By using a static member **print the number of objects** created for Rectangle and for Square**.**

2. **Define and implement the class Calendar** with the following members: Year, Month, Day, NameOfDay (enumerate from Monday to Sunday) by using

enum Days{Monday,Tuesday,Wednesday,Thursday,Friday,Saturday, Sunday};
Days Day;

and as member functions :

-a default constructor, an explicit one, a copy constructor
-getting the current date
-modify the current date
-increment the current date
-print the current date

Define a friend function in order to read the current date.

HOMEWORK:

1. Create and implement a class Triangle which allows us to do some operations as follows: detecting the types of the triangles: isosceles triangle, equilateral, rectangular, how many degrees have the angles in each case, apply at least 3 theorems (Pythagoras, Thales, catheters Theorem, Theorem height,...). Use: default constructor, explicit, copy constructor, destructor, static and friend functions, static and friend members, const, mutable, the pointer *this*.

Use your imagination and create a friend class for the Triangle class which uses some members of the Triangle class.