C++

Friend-Function & Class

1. Friend - Problem - I

- **Write a C++ program which is capable to add to complex number.**
- **Write a C++ program which is capable to add to complex number using class.**
- **Write a C++ program which is capable to add to complex number using friend function.**

1. Friend - Function

- **A friend function of a class is a non-member function of the class that can access its private and protected members.**

- **To declare an external function as a friend of the class, you must include function prototype in the class definition. The prototype must be preceded with keyword friend.**

1. Friend - Function (Syntax)

```
class class_name
{
        -------
        -------
        friend return_type function_name(list of arguments);
        -------
};

return_type function_name(list of arguments)
{
        -------
        -------
}
```

1. Friend - Function

- **Friend function is a normal external function that is given special access privileges.**

- **It is defined outside that class' scope. Therefore, they cannot be called using the '.' or '->' operator. These operators are used only when they belong to some class.**

1. Friend - Function

- **While the prototype for friend function is included in the class definition, it is not considered to be a member function of that class.**

- **The friend declaration can be placed either in the private or in the public section.**

- **A friend function of the class can be member and friend of some other class.**

1. Friend - Function

- **Since friend functions are non-members of the class, they do not require this pointer. The keyword friend is placed only in the function declaration and not in the function definition.**

- **A function can be declared as friend in any number of classes.**

- **A friend function can access the class's members using directly using the object name and dot operator followed by the specific member.**

1. Friend - Function

```
class Distance
{
        private:
                int meter;
                int km;
                friend float convert(Distance d);
        public:
                void get_data()
                {
                        cout<<"Enter kms and
meters : ";
                        cin>>km>>meter;
                }
};
float convert(Distance d)
{
        return ((d.km*1000)+d.meter);
}
```

```
int main()
{
        Distance d;
        d.get_data();
        cout<<"Distance : "<<convert(d)<<"m";

        return 0;
}
```

1. Friend - Class

A friendship is not transitive. This means that friend of a friend is not considered a friend unless explicitly specified. For example, if class A is a friend of class B and class B is a friend of class C, then it does not imply—unless explicitly specified—that class A is a friend of class C.

When a class is declared a friend class, all the member functions of the friend class become friend functions.

1. Friend - Class

- **A friend class is one which can access the private and/or protected members of another class.**

- **To declare all members of class A as friends of class B, write the following declaration in class A.**

friend class B;

```
class B;
class A
{   friend class B;
    data members
    member functions
};
class B
{   data members
    member functions
};
```

1. Friend - Class

- **Similar to friend function, a class can be a friend to any number of classes.**

- **When we declare a class as friend, then all its members also become the friend of the other class.**

- **You must first declare the friend becoming class (forward declaration).**

- **A friendship must always be explicitly specified. If class A is a friend of class B, then class B can access private and protected members of class B. Since class B is not declared a friend of class A, class A cannot access private and protected members of class B.**

1. Friend - Class

```
int main() {
    ClassB objectB;
    cout << "Sum: " << objectB.add();
    return 0;
}
```

```
#include <iostream>
using namespace std;

class ClassB;

class ClassA {
    private:
        int numA;

        friend class ClassB;

    public:
        ClassA() : numA(12) {}
};
```

```
class ClassB {
    private:
        int numB;

    public:
        ClassB() : numB(1) {}
    int add() {
        ClassA objectA;
        return objectA.numA + numB;
    }
};
```

1. Friend - Problem - II

- **Write a C++ program which is capable to create two different subject classes like Math & English. Each class contains two data member one is roll and another is age. You need to compare the result using one user defined function.**