Friend functions, pointers to class members

Friend function and classes: **This is a flow** (because in OOP private and protected members cannot be accessed).A friend function can access private and protected members of other class in which it is declared as a friend.

Eg: friend class:
```
Class A{
Private: int a;
Public: A(){
a=0;
}
Friend class B;
};
Class B{
Private: int b;
Public: void showA(A&x){//pass by reference
Cout<<x.a;
  }
};
```

Friend function is not 2-way property i.e., Class b can access private and protected members of class A but class A cannot access private and protected members of class B.

Eg – friend function : It can be given special grant to access private and protected members. A friend fuction can be:

- A method of another class.
- A global function.

```
Class A{

Private:int a;

Public: friend void B::showA(A& x);//only showA function of class B
can access private or protected members of class A

};


Class B{

Private:int b;

Public: void showA(A& x){

Cout<<x.a;

      }

};
```

Constant member function : An object declared as constant cannot be modified and hence, can invoke only const functions as these functions ensure not to modify the object.

A const object can be created by prefixed the const keyword to the object declaration.

Any attempt to change the data member of const objects results in a compile-time error.

Whenever an object is declared as const, it needs to be initialized at the time of declaration. However, the object initialization while declaration is possible only with the help of constructors.

When a function is declared as const, it can be called on any type of object, const object as well as non-const objects.

Const functions are not allowed to modify the object on which they are called.

Defining a pointer of class type:

We can define pointer of class type, which can be used to point to class objects:

```
Class obj;

Class* ptr;

Ptr = &obj;// is equal *ptr=obj;

Cout<<obj.a;

Cout<<ptr->a;
```

Pointers to data members of class:

We can use pointer to point the class's data members .

Synatx: datatype class_name::*pointer_name;

Syntax for assignment:

Poiter_name=&class_name::datamember_name;

Eg:

```
Class Data{

Public: int a;

Void print(){

Cout<<"a is"<<a;

  }
};
Data d, *dp;
Dp=&d;
Int Data::*ptr=&Data :: a;
d.*ptr=10;// is equal to d.a cz a=*ptr;
d.print();
dp->*ptr=20;
dp->print();
```

Pointers to member functions of class:

Syntax:
return_type(class_name::*ptr_name)(argument_list)=&clas_name::function_name.

Eg:

```
Class Data{
Public: int f(float a){
Return 1;
  }
};
Int(Data::*fp1)(float)=&Data::f;
```