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

class Complex{
   int real, imaginary;
   public:
      void getData(){
         cout<<"The real part is "<< real<<endl;
         cout<<"The imaginary part is "<< imaginary<<endl;
      }

      void setData(int a, int b){
         real = a;
         imaginary = b;
      }

};
int main(){
   Complex *ptr = new Complex;
   (*ptr).setData(1, 54); //is exactly same as
   (*ptr).getData(); //is as good as

   return 0;
}



#include<iostream>
using namespace std;
class Time
{
          short int hh, mm, ss;
```

```
        public:
                Time()
                {
                        hh = mm = ss = 0;
                }
                void getdata(int i, int j, int k)
                {
                        hh = i;
                        mm = j;
                        ss = k;
                }
                void prndata(void)
                {
                        cout<<"\nTime is "<<hh<<":"<<mm<<":"<<ss<<"\n";
                }
};
int main()
{

        Time T1, *tptr;
        cout<<"Initializing data members using the object, with values 12, 22, 11\n";
        T1.getdata(12,22,11);
        cout<<"Printing members using the object ";
        T1.prndata();
        tptr = &T1;
        cout<<"Printing members using the object pointer ";
tptr->prndata();
        cout<<"\nInitializing data members using the object pointer, with values 15, 10, 16\n";
        tptr->getdata(15, 10, 16);
        cout<<"printing members using the object ";
        T1.prndata();
```

```
        cout<<"Printing members using the object pointer ";
        tptr->prndata();
        return 0;
}


#include<iostream>
using namespace std;

class Date
{
   private:
      short int dd, mm, yy;

   public:
      Date() //constrctor:
        {
          dd = mm = yy = 0;
        }

      void getdata(int i, int j, int k)
        {
          dd = i;
          mm = j;
          yy = k;
        }


      void prndata(void)
        {
          cout<<"\nData is "<<dd<<"/"<<mm<<"/"<<yy<<"\n";
```

```
        }
};

int main()
{
   Date D1; //simple object having type Data:
   Date *dptr; //Pointer Object having type Date:

   cout<<"Initializing data members using the object, with values 19, 10, 2016"<<endl;
   D1.getdata(19,10,2016);

   cout<<"Printing members using the object ";
   D1.prndata();

   dptr = &D1;
   cout<<"Printing members using the object pointer ";
   dptr->prndata();

   cout<<"\nInitializing data members using the object pointer, with values 20, 10, 2016"<<endl;
   dptr->getdata(20, 10, 2016);
   cout<<"printing members using the object ";
   D1.prndata();

   cout<<"Printing members using the object pointer ";
   dptr->prndata();

   return 0;
}



#include<iostream>
```

```
using namespace std;
class Simple
{
   public:
   int a=10;
};

int main()
{
   Simple obj;
   Simple* ptr;   // Pointer of class type
   ptr = &obj;

   cout << obj.a<<"\n";
   cout << ptr->a<<"\n";  // Accessing member with pointer
}
```

```
#include<iostream>
using namespace std;
class Data
{
                public:
   int a;
   void print()
   {
      cout << "a is "<< a<<"\n";
   }
};

int main()
```

```
{
   Data d, *dp;
   dp = &d;    // pointer to object

   int Data::*ptr=&Data::a;   // pointer to data member 'a'



   d.*ptr=10;
   d.print();

   dp->*ptr=20;
   dp->print();
}



#include <iostream>
#include <string>
using namespace std;
class student
{
private:
        int rollno;
        string name;
public:
        student()
        {rollno=0;
                        name=" ";
                        }
        student(int r, string n)
        {
                        rollno=r;
```

```
                        name=n;
                        }
        void get()
        {
                cout<<"enter roll no";
                cin>>rollno;
                cout<<"enter  name";
                cin>>name;
        }
        void print()
        {
                cout<<"roll no is:"<<rollno<<"\n";
                cout<<"name is:"<<name<<"\n";
        }
};
int main ()
{
        student *ps=new student;
        (*ps).get();
        (*ps).print();

        return 0;
}
```

```
#include <iostream>
using namespace std;
class BaseClass {
public:
  void disp(){
    cout<<"Function of Parent Class";
  }
};
class DerivedClass: public BaseClass{
public:
  void disp() {
    cout<<"Function of Child Class";
  }
};
int main() {

   BaseClass obj;
  obj.disp();
  return 0;
}
```