```
#include<iostream>
using namespace std;
// Base class
class Vehicle {
 public:
    string brand = "Ford";

              void honk() {
      cout << "Tuut, tuut! \n" ;
    }
};

// Derived class
class Car: public Vehicle {
  public:
    string model = "Mustang";
};

int main() {
  Car myCar;
  myCar.honk();
  cout << myCar.brand + " " + myCar.model;
  return 0;
}

#include<iostream>
using namespace std;
// Base class
class Employee {
  protected: // Protected access specifier
    int salary;
```

```
};



// Derived class
class Programmer: public Employee {
  public:
    int bonus;

    void setSalary(int s) {
      salary = s;
    }
    int getSalary() {
      return salary;
    }
};

int main() {
  Programmer myObj;
  int p,t;
  myObj.setSalary(50000);
  myObj.bonus = 15000;
  cout << "Salary: " << myObj.getSalary() << "\n";
  cout << "Bonus: " << myObj.bonus << "\n";
  cout<<"Enter value of p and t\n";
  cin>>p>>t;
  myObj.setSalary(p);
  myObj.bonus = t;
  cout << "Salary: " << myObj.getSalary() << "\n";
  cout << "Bonus: " << myObj.bonus << "\n";
  return 0;
}

#include<iostream>
using namespace std;
// Base class
class MyClass {
  public:
              int g,k;
              float r;
    void myFunction() {
      cout<<"Enter value of g,k and r\n";
      cin>>g>>k>>r;
      cout<<"g:"<<g<<"\n"<<"k:"<<k<<"\n"<<"r"<<r<<"\n";
                cout << "Some content in parent class.\n" ;
    }
};

// Another base class
class MyOtherClass {
  public:
    void myOtherFunction() {
      cout << "Some content in another class." ;
    }
};

// Derived class
class MyChildClass: public MyClass, public MyOtherClass {
};

int main() {
  MyChildClass myObj;
  myObj.myFunction();
  myObj.myOtherFunction();
  return 0;
}

#include<iostream>
using namespace std;
// Base class (parent)
class MyClass {
  public:
    void myFunction() {
      cout << "Some content in parent class.\n" ;
    }
};

// Derived class (child)
class MyChild: public MyClass {
        public:
                int s;
                void b()
                {
                        cout<<"Enter value for s: \n";
                        cin>>s;
                        cout<<"s:"<<s<<"\t"<<"\n";
                }
};

// Derived class (grandchild)
class MyGrandChild: public MyChild {
};
```

```
int main() {
  MyGrandChild myObj;
  myObj.b();
  myObj.myFunction();
  return 0;
}

// C++ program to demonstrate hierarchical inheritance

#include <iostream>
using namespace std;

// base class
class Animal {
   public:
    void info() {
        cout << "I am an animal." << endl;
    }
};

// derived class 1
class Dog : public Animal {
   public:
    void bark() {
        cout << "I am a Dog. Woof woof." << endl;
    }
};

// derived class 2
class Cat : public Animal {
  public:
   void meow() {
       cout << "I am a Cat. Meow." << endl;
   }
};

int main() {
   // Create object of Dog class
   Dog dog1;
   cout << "Dog Class:" << endl;
   dog1.info();  // Parent Class function
   dog1.bark();

   // Create object of Cat class
   Cat cat1;
   cout << "\nCat Class:" << endl;
   cat1.info();  // Parent Class function
   cat1.meow();

   return 0;
}

#include<iostream>
using namespace std;
class Base
{
		public:
		virtual   void show()
			{
				cout<<"Base Class\t";
```

```
                }
};
class Derived:public Base
{
        public:
                void show()
                {
                        cout<<"Derived Class";
                }
};
int main()
{
        Base *b;
        Derived d;
        b=&d;
        b->show();

}
```

```
#include<iostream>
using namespace std;
class Base
{
        int b;
        protected:
                int a;
        public:
                void show()
                {   b=55;
                a=20;
                cout<<"a="<<a<<"\t";
                        cout<<"Base Class\t";
                        cout<<"b="<<b<<"\t";


                }
};
class Derived:public Base
{
        public:
                void show()
                {
                        cout<<"Derived Class";
                }
};
int main()
{
        Base b;
        Derived d;
        b.show();
        d.show();
}
```