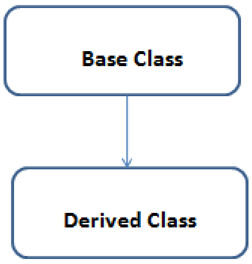
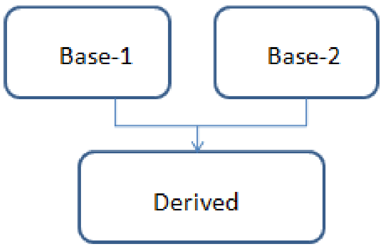
Q.1 Define Inheritance. List different types of inheritances with suitable diagram.
A. The mechanism of deriving new class from an old/existing class is called inheritance. Inheritance is the process by which objects of one class acquired the properties of objects of another classes.
Syntax:
class derived-class-name: visibility-mode base-class-name
{
------//
-----// members of derived class
-----//
};

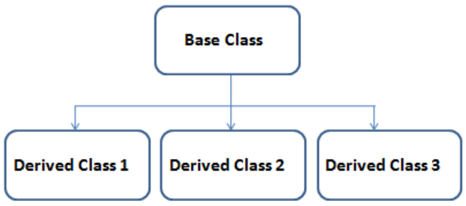
Types of inheritance:
1) Single inheritance: In single inheritance, a derived class is derived from only one base class.



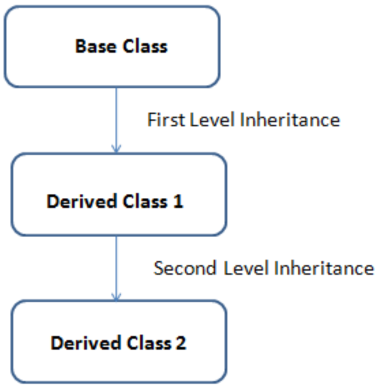
2) Multiple inheritance: In multiple inheritance, derived class is derived from more than one base classes.



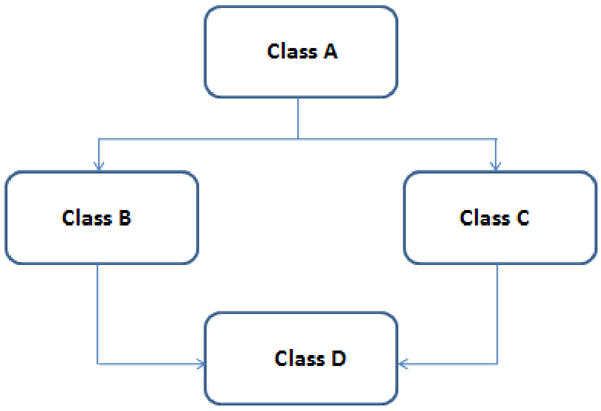
3) Hierarchical inheritance: In hierarchical inheritance, more than one derived classes are derived from single class.



4) Multilevel inheritance: In multilevel inheritance, a derived class is derived from a derived class (intermediate base class) which in turn derived from a single base class.



5) Hybrid inheritance: Hybrid inheritance is a combination of single, multiple, multilevel and hierarchical inheritance.



Q.2 Explain various visibility modes of class.
A. In inheritance whenever the derived class inherits from the base class than which of the member of the parent class can be accessible in the child class is controlled by the visibility mode. By default visibility mode is always set to private.
1. Public Visibility mode: If we derive a subclass from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in the derived class.
2. Protected Visibility mode: If we derive a subclass from a Protected base class. Then both public member and protected members of the base class will become protected in the derived class.
3. Private Visibility mode: If we derive a subclass from a Private base class. Then both public member and protected members of the base class will become Private in the derived class.

Q.3 Explain single inheritance with example.
A. Single inheritance is one type of inheritance in which the derived class inherits only one base class. It provides reusability by allowing the derived class to inherit the features of the base class using objects.

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

```
    cout << "This show is inside the second class which
is derived from parent class" << endl;
```

```
class First {
  public: void display() {
    cout << "This display is inside the first class" <<
endl;
  }
};
class Second: public First {
  public: void show() {
```

```
  }
};
int main() {
  Second s;
  s.display();
  s.show();
}
```

Q.4 Demonstrate use of Multilevel Inheritance with suitable example.
A.

```
#include <iostream>
using namespace std;
class base
{
  public:
          int x=5;
};
class derive1: public base
{
  public:
          int y=4;
};
```

```
class derive2: public derive1
{
  private: int z=2;
  public:
          void product() {
            cout << "Product = " << x * y * z;
          }
};
int main() {
  derive2 a;
  a.product();
  return 0;
}
```

Q.5 Demonstrate use of Multiple Inheritance with suitable example.
A.

```
#include <iostream>
using namespace std;
class Base_class {
  public:
    void display() {
      cout << " It is the first function of the Base class "
<< endl;
    }
};
class Base_class2 {
  public:
    void display2() {
      cout << " It is the second function of the Base class
" << endl;
    }
};
```

```
class child_class: public Base_class, public Base_class2
{
  public:
        void display3()
        {
          cout << " It is the function of the derived class
" << endl;
        }

};
int main() {
  child_class ch;
  ch.display();
  ch.display2();
  ch.display3();
}
```

Q.6 Give difference between multilevel and multiple inheritance.
A.

Multilevel Inheritance	**Multiple Inheritance**
When a class inherits other class and then that inherited class further inherited by other class, this type of inheritance is called multilevel inheritance. In multilevel inheritance, sub class contains the properties of the base class as well as properties of base class of its own base class.	When a class inherits the properties and behaviours of more than one class then this type of inheritance is called multiple inheritance. In multiple inheritance, there is only one sub class but more than one super class.

Q.7 Explain constructor & destructor in derive class using suitable example.
A. In inheritance, When an object of derived class is created then constructor of derived class get executed and then it calls the constructor of base class.
If there is no default constructor present in parent class then not only we have to create constructor in child class but also we will have to call the constructor of parent class.

In order to call parameterized constructor of parent class, we need to create a constructor in child class and also, we will have to call parameterized constructor with the help of child class constructor.

```
#include<iostream>
using namespace std;
class base
{
  int x;
  public:
    base(int a)
  {
    x = a;
    cout << "Constructor in base x=" << x;
  }
};
class derived: public base
{
  int y;
  public:
    derived(int a, int b): base(a)
  {
    y = b;
    cout << "\nConstructor in derived y=" << y;
  }
};
int main()
{
  derived ob(2, 3);
  return 0;
}
```

Q.8 Explain Hierarchical inheritance with example.
A. In hierarchical inheritance all the derived classes have common base class. The base class includes all the features that are common to derived classes.

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

class A
{
    public:
            int x=5, y=8;
};
class B : public A
{
    public:
            void product()
            {
                cout << "\nProduct= " << x * y;
            }
};
class C : public A
{
    public:
            void sum()
            {
        cout << "\nSum= " << x + y;
            }
};
int main()
{
    B obj1;
    C obj2;
    obj1.product();
    obj2.sum();
    return 0;
}
```

Q.9 Why we need to define Virtual base class in Hybrid inheritance? Explain with example.
A. Virtual base classes are used in inheritance to prevent multiple “instances” of a given class appearing in an inheritance hierarchy when using multiple inheritances.

```
#include <iostream>
using namespace std;
class A {
public:
    void show()
    {
        cout << "Hello from A \n";
    }
};
class B : public virtual A {
};
}
class C : public virtual A {
};
class D : public B, public C {
};
int main()
{
    D object;
    object.show();
```

Q.10 Explain abstract class with example.
A. An abstract class a class that has at least one pure virtual function (i.e., a function that has no definition). The classes inheriting the abstract class must provide a definition for the pure virtual function; otherwise, the subclass would become an abstract class itself. Abstract classes are essential to providing an abstraction to the code to make it reusable and extendable.

```
#include<iostream>
using namespace std;
class Base
{
int x;
public:
	virtual void fun() = 0;
	int getX() { return x; }
};
```

```
class Derived: public Base
{
	int y;
public:
	void fun() { cout << "fun() called"; }
};
int main(void)
{
	Derived d;
	d.fun();
	return 0;
}
```

Q.11 What is ambiguity in multiple inheritance?
A. In multiple inheritances, when one class is derived from two or more base classes then there may be a possibility that the base classes have functions with the same name, and the derived class may not have functions with that name as those of its base classes. If the derived class object needs to access one of the similarly named member functions of the base classes then it results in ambiguity because the compiler gets confused about which base's class member function should be called.

Q.12 Explain method overriding with example.
A. Function overriding is a concept by which you can define a function of the same name and the same function signature (parameters and their data types) in both the base class and derived class with a different function definition. It redefines a function of the base class inside the derived class, which overrides the base class function. Function overriding is an implementation of the run-time polymorphism. So, it overrides the function at the run-time of the program.

```
#include <iostream>
using namespace std;
class parent_class
{
public:
   virtual void print()
   {
	cout << "\nThis is print() method of
BaseClass";
   }
};
```

```
class derived_class : public parent_class
{
public:
   void print()
   {
      cout << "\nThis is print() method of the Derived
Class";
   }
};
int main()
{
   derived_class obj;
   obj.print();
}
```

Q.13 Write advantages and disadvantages of Inheritance.
A. **Advantages:**
Inheritance promotes reusability. When a class inherits or derives another class, it can access all the functionality of inherited class.
Reusability enhanced reliability. The base class code will be already tested and debugged.
As the existing code is reused, it leads to less development and maintenance costs.
Inheritance makes the sub classes follow a standard interface.
Inheritance helps to reduce code redundancy and supports code extensibility.
Inheritance facilitates creation of class libraries.

Disadvantages:
Inherited functions work slower than normal function as there is indirection.
Improper use of inheritance may lead to wrong solutions.
Often, data members in the base class are left unused which may lead to memory wastage.
Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.