Q- 1 What is inheritance? Explain it with example.

"The capability of a class to derive properties and characteristics from another class is called **Inheritance**. "

Inheritance is one of the most important features of Object Oriented Programming in C++.

The syntax of inheritance is the following :

```
class  derived_class_name : access-specifier base_class_name
{

        //    body ....

};
```

In the above syntax :

- **class: Class is the** keyword to create a new class
- **Derived_class_name**: It is the name of the new class, which will inherit the base class
- **access-specifier**: Specifies the access mode which can be either of private, public or protected.
- If neither is specified, private is taken as default.
- **base-class-name**: It is the name of the base class.

Example of inheritance :

The example of inheritance is the following :

```
#include<iostream>
#include<conio.h>

using namespace std;
class a
{

    public:
    a()
        {
                cout<<"hello"<<endl;
    }
};
class b : public a
{
        public:
        b()
        {
                cout<<"hi";
        }

};
int main()
{
        b p;

        getch();
}
```

Q- 2 What is inheritance? Explain it types with example.

"The capability of a class to derive properties and characteristics from another class is called **Inheritance**. "

Inheritance is one of the most important features of Object Oriented Programming in C++.

There are 5 types of inheritance is the following :

1. **Single inheritance**
2. **Multilevel inheritance**
3. **Multiple inheritance**
4. **Hierarchical inheritance**
5. **Hybrid inheritance**

1) Single Inheritance :

"Single Inheritance is the process in which child class inherits the properties and characteristics of only one base class."

.

i.e. one base class is inherited by one derived class only.

The syntax of SINGLE inheritance is the following :

```
class  derived_class_name : access-specifier base_class_name
{

       //    body ....

};
```

In the above syntax :

- **class: Class is the** keyword to create a new class
- **Derived_class_name**: It is the name of the new class, which will inherit the base class
- **access-specifier**: Specifies the access mode which can be either of private, public or protected.
- If neither is specified, private is taken as default.
- **base-class-name**: It is the name of the base class.

Example of inheritance :

The example of inheritance is the following :

```
#include<iostream>
#include<conio.h>

using namespace std;
class a
{

   public:
   a()
      {
            cout<<"hello"<<endl;
   }
};
class b : public a
{
      public:
      b()
      {
            cout<<"hi";
      }

};
int main()
{
      b p;
```

```
    getch();
}
```

Multilevel Inheritance :

“Multilevel Inheritance is the process in which child class inherits the properties and characteristics of one base class.”

Child class also make a base class it is further inherited by another class.

It is known as multi-level inheritance.”

For example :

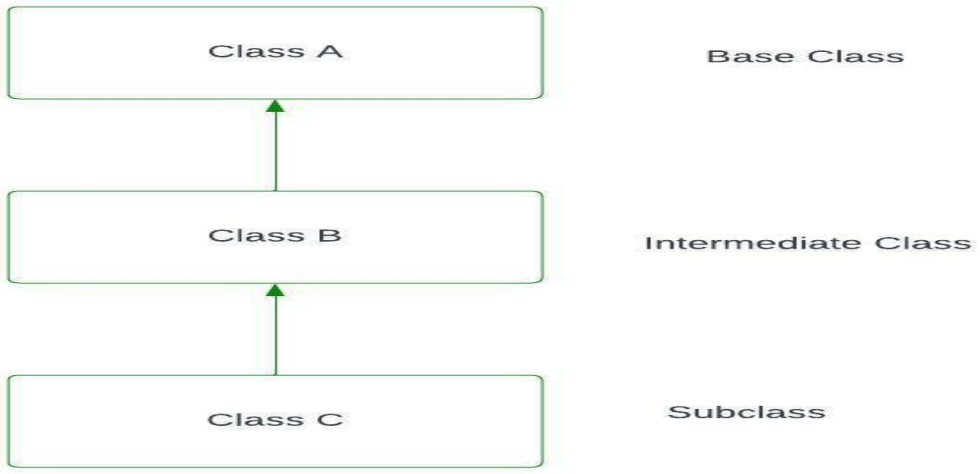
if we take Grandfather as a base class then Father is the derived class that has features of Grandfather.

Then Child is the also derived class that is derived from the sub-class Father which inherits all the features of Father.

Another Example:

```
Base class-> Wood,
Intermediate class-> furniture,
subclass-> table.
```

Block Diagram of Multilevel Inheritance



In above diagram class B inherits property from class A and class C inherits property from class B.

Syntax:

```
class A // base class
{
    ..........
};
class B : access_specifier A // derived class
{
    ..........
} ;
class C : access_specifier B // derived from derived class B
{
    ..........
} ;
```

Example :

```
#include<iostream>
#include<conio.h>
using namespace std;
class multilevel
{
     public :
          int n;
     void input()
     {
        cout<<"n=";
        cin>>n;
     }
};
class level1:public multilevel
{
     public :
          int m;
```

```
    void input2()
    {
        cout<<"m=";
        cin>>m;
    }
};
class level2 : public level1
{
     public:

        void display()
        {
            if(m>n)
               cout<<m<<" is big";
            else
               cout<<n<<" is big";
         }
};

int main()
{
    level2 p;

    p.input() ;
    p.input2() ;
    p.display();

    getch();

}
```

Multiple Inheritance :

“Multiple Inheritance is the process in which child class inherits the properties and characteristics of more then one base class.”

It is known as multi-ple inheritance.”

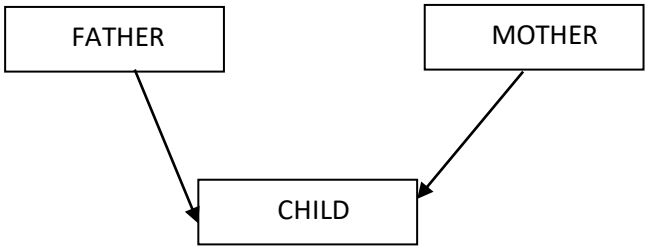
For example :

if we take father and mother as a base class then child is the derived class that has features of both father and mother.

Another Example:

```
Base class-> Jungle,
Base class-> wood,
subclass-> table.
```

Block Diagram of Multilevel Inheritance



In above diagram class CHILD CLASS inherits property from class FATHER CLASS and MOTHER CLASS.

Syntax:

```
class A
{
... .. ...
};
class B
{
... .. ...
};
class C: public A,public B
{
... ... ...
};
```

EXAMPLE :

```
#include<iostream>
#include<conio.h>
using namespace std;

class father
{
	public :
		int a;
	void input()
	{
		cout<<"a=";
		cin>>a;
	}

};
class mother
{
	public :
		int b;
	void input1()
	{
		cout<<"b=";
		cin>>b;
	}

};
class child:public father,public mother
{
	public:
		int sum;

		void output()
		{
			sum= a+b;
			cout<<"sum="<<sum;
		}
};

int main()
{
	 child p;
```

```
        system("cls");
        p.input();
        p.input1();
        p.output();

        getch();
}
```

HIERARCHICAL INHERITANCE :

Hierarchical Inheritance is the process in which a derived class (child class) inherits the property (data member and member functions) of the Base class (parent class).

In Hierarchical inheritance, more than one sub-class inherits the property of a single base class.

There is one base class and multiple derived classes.

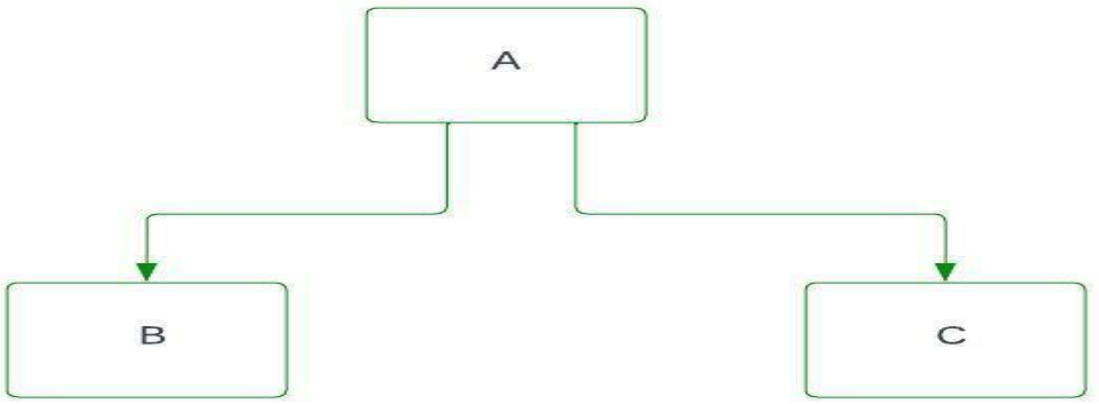
Several other classes inherit the derived classes as well.

Hierarchical structures thus form a tree-like structure.

It is similar to that, mango and apple both are fruits; both inherit the property of fruit.

Fruit will be the Base class, and mango and apple are sub-classes.

BLOCK DIAGRAM :



In above diagram , Class A is a Base class,

B is a subclass inherited from class A, and C is a subclass it also inherits from class A.

SYNTAX :

```
Class A
{
   .............
};
Class B: access_specifier A
{
   .........
};
Class C: access_specifier A
{
  ..............
};
```

EXAMPLE

```
#include<iostream>
#include<conio.h>
using namespace std;

class father
{
        public :
              int a;
        void input()
```

```
        {
                cout<<"a=";
                cin>>a;
        }

};
class mother : public father
{
        public :
                int b;
        void input1()
        {
               b=a;
                cout<<"b="<<b;

        }

};
class child : public father
{
       public:
               int sum;

               void output()
               {
                       Sum=a;
                       cout<<"sum="<<sum;
               }
};
int main()
{
       child p;
        mother k;

        system("cls");
        p.input();
        p.input1();
         k.input();
        k.output();

        getch();
}
```

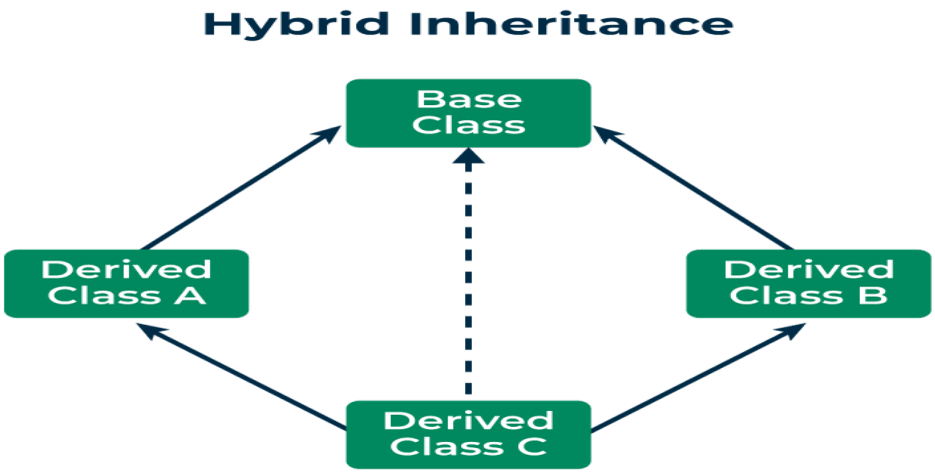
HYBRID INHERITANCE :

"Hybrid Inheritance is the process in which multiple types of inheritance are combined within a single class hierarchy, enabling a varied and flexible structure of classes."

In hybrid inheritance, within the same class, we can have elements of **single inheritance**, **multiple inheritance**, **multilevel inheritance**, and **hierarchical inheritance**.

Hybrid inheritance is a complex form of inheritance in object-oriented programming (OOP).

BLOCK DIAGRAM :



In above diagram, Class A and Class B inherits the properties of Base class i.e hierarchical inheritance .

Class C inherits the properties of Class A and Class B i.e Multiple inheritace.

Example :

```
#include<iostream>
#include<conio.h>
using namespace std;

class father
{
                    public :
                       int a;
                    void input()
                    {
                       cout<<"a=";
                       cin>>a;
                    }

};
class mother : public father
{
                    public :
                       int b;
                     mother()
```

```
                    {
                      b=5;
                    }

};
class child : public father
{
                    public:
                       int sum;

                       child()
                       {
                            sum=10;
                       }
};
class child2:public mother, public child
{
    public:
        int mul;

            void display()
            {
                    mul = sum * b;
                    cout<<"mul="<<mul;
            }
};
int main()
{
```

```
        child2 m;

                        system("cls");

                        m.display();

                        getch();
}
```