Assignment - 13

Q.1. What is the use of copy constructor.

→ • A copy constructor is a member function that initializes an object using another object of the same class.

• In simple terms, contructor which creates an object by initializing it with an object of same class, which has been created previously is known as a pot copy constructor.

• copy constructor is used to initialize the members of a newly created object by copying the members of an already existing object.

Example -

```
Sample ( Sample & ref)
{
    i = ref.i ;
    j = ref.j ;
}
```

Q.2. When constructor and destructor gets called ?

—> • constructors and destructors are considered as a special functions.

• constructor is a function which gets automatically called when we create object of that class.

• compiler will call constructor before allocating memory for object.

• Destructor is is special function which gets automatically called before deallocating memory of object.

Example -

```
class Demo{
    Public :
        int i, j;
    Demo c) {
        cout << " Inside Default constructor \n ";
            i = 0 ;
            j = 0 ;
            }

        Demo ( int A, int B) {
            cout << " Inside Parameterized constructor \n ";
                i = A ;
                j = B ;
                }
```

```
Demo(Demo& ref) {
    cout << " Inside copy constructor \n";
        i= i. ref ;
        j = j. ref;
          }
    ~ Demo ()
    {
        cout << " Inside Destructor \n";
     }
};

int main()
{
    Demo obj1;                  // Default
    Demo obj2(10, 20);          // parameterized
    Demo obj3 (obj2);           // copy constructor
    return 0;

}
```

Q.3. What are the rules which has to followed writing constructors and destructor?

—> • Following rules should follow while writing constructors & destructor :

1. Name of constructor should be same as class name.

2. All the constructor & destructor should be inside inside - public access specifier.

3. There should not be any return value from constructors & destructor.

4. There is no need of explicit call to the constructors & destructor.

5. If we create an object without passing parameter then default constructor gets called.

6. If we create an object by passing some parameter then parameterized constructor gets called.

7. If we create an object by passing another object then copy constructor gets called.

Q.4. What is mean by this pointer?

-> • This pointer is used to call any non-static member function. We need object of that class.

• By using the object and direct accessing operator (.) we call any non-static method of a class.

• When we use that caller object, internally compiler will sends the address of that object as a first hidden implicit parameter.

• That address gets store inside the first implicit argument of the function. and that argument is called as this pointer.

• This is considered as keyword in c++ and Java.

e.g -

```
class Demo{ public:
        void fun( int a, int b){
        } //
};
int main(){
Demo obj;
obj.fun (10, 20);
}
```

Address of object.

← Hidden argument

internally - fun (&obj, 10, 20)

Q.5. What is the prototype of this pointer?

—> eg.

```
class Demo
{
    public :
        // void Add (Demo * this , int No1 , int No2)
        void Add (int No1 , int No2)
        {
            cout << "Inside Add" ;
        }
};
int main()
{
    Demo obj1;
    obj1.Add (10,20);

    // obj1.Add (10,20);

    // Add (&obj , 10, 20);

    return 0;
}
```

Q.6. What is mean by inheritance? Explain types of Inheritance.

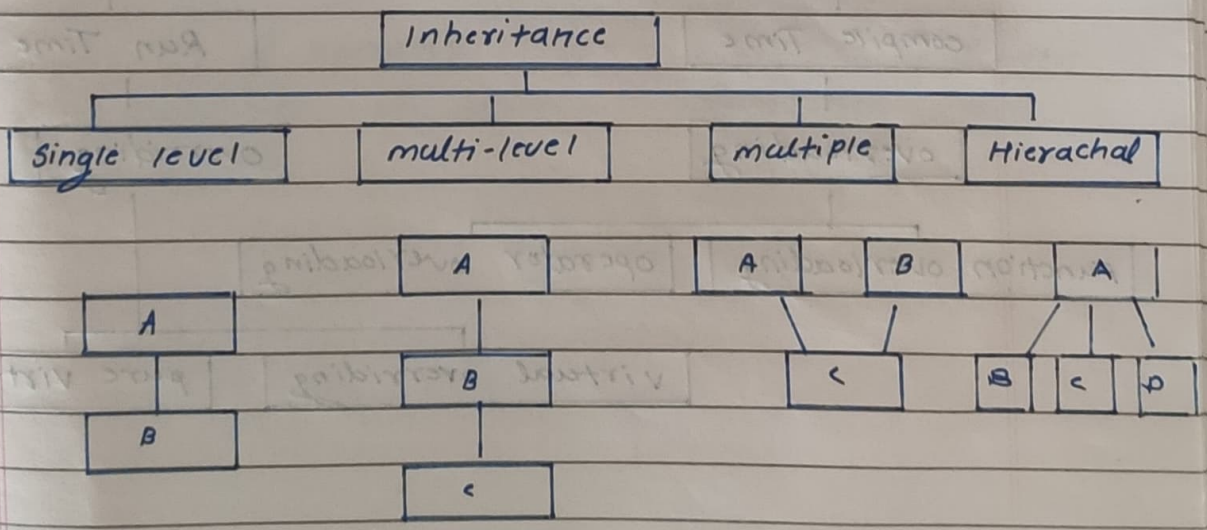
-> • Inheritance is considered as one of the OOP's paradigm of c++ and Java.

• In simple words Inheritance defined as re-usability.

• By using concept of Inheritance one class can acquire properties (characteristics & behaviour) of another class.

• According to the structural Layout there are three major types of Inheritance:

1. Single level Inheritance
2. Multi level Inheritance
3. multiple Inheritance
4. Hierarchical Inheritance.



Q.7. What is mean by polymorphism? Explain its types.

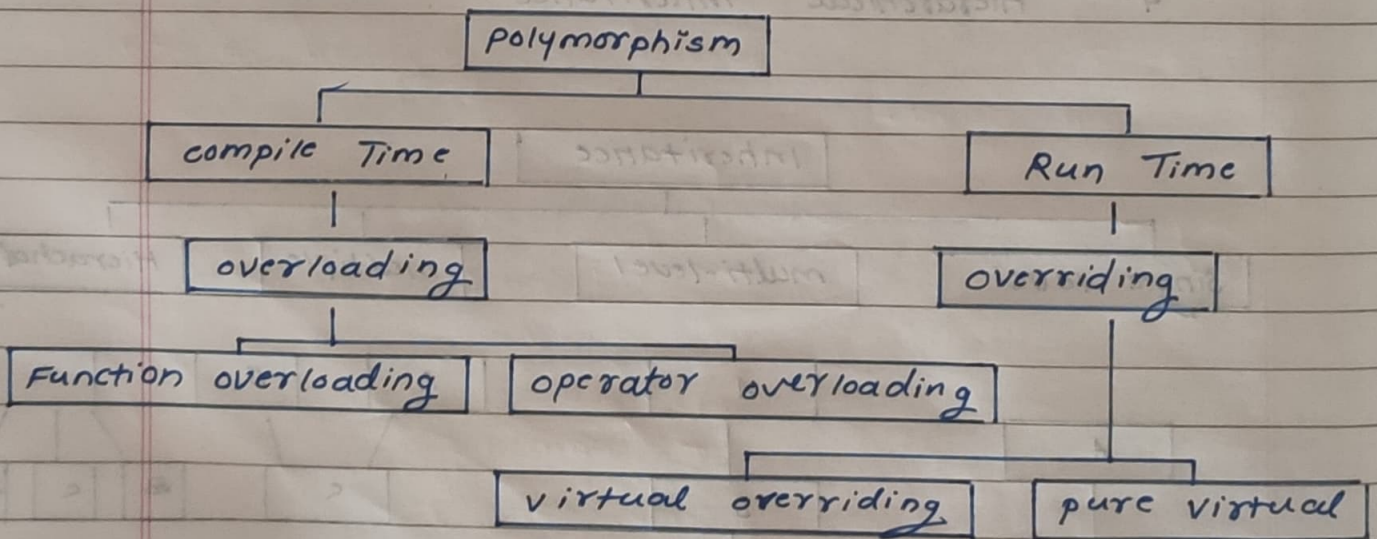
—> • The term polymorphism is available in every oop language with some changes.

• The concept of polymorphism is defined as single name and multiple behaviours.

• Poly means many and morphism means forms hence many forms is considered as polymorphism of an method.

• Polymorphism is divided into two types:

1. Compile Time polymorphism
2. Run Time Polymorphism



Q.8. What is the use of [copy constructor] & operator in case of copy constructor?

→
- When we use & operator after = assignment operator then it is considered as address of operator.
- If & operator is used before = assignment operator then it is considered as reference operator.
- In case of copy constructor we use the & operator in argument to define copy of other function.
- & operator is used in between method name and copy constructors other name

for e.g
```
Demo (Demo &ref) {
//
}
main() -> Demo obj1(&obj2);
```
- The & operator is used to send the reference of another object to other object.

Q.g. Why the name of constructor & Destructor is same as class name?

—> Every class object is created using the same new keyword, so it must have information about the class to which it must create an object. for this reason, the constructor name should be same as the class name.

```
class Demo {
        Public:
            int a,b;
        Demo() {
            cout << "inside Default Demo";
            a = 0;
            b = 0;
        }
```

Q.10. What is mean by function overloading?

→ • function overloading is considered as compile time polymorphism.

• Under compile time polymorphism we use the concept of overloading.

• In case of function overloading we can define multiple function in our class with same name and with different prototypes.

• Example -

```
class Demo {
        public :
// same name different prototype
    int Add ( int A , int B )
    {
        int Ans = 0;
        Ans = A+B;
        return Ans;
    }
    int Add ( int A , int B, int C)
    {
        int Ans = 0;
        Ans = A+B+C;
        return Ans;
    }
};
int main () {
```

```
    Demo obj;
    int Ans1 = 0;
    obj.Add(10,20);   cout << "Addition:" << Ans1 << "\n";
Ans1 = obj.Add(10,20,30);
    cout << "Addition is :" << Ans1 << "\n";
}   return 0;
```