Qus 1:- What is constructor?

Ans:- 1. constructor is a special type of function which has the same name as the class anme.

2. constructor is being automatically call at the time of object declartion or creation.

3. Constructors don't have return type is used for initializing the objects

4. It must be placed in **public section** of class.

5. If we do not specify a constructor, C++ compiler generates a default constructor for object **(expects no parameters and has an empty body).**

6. Constructors can be defined inside or outside the class declaration

using scope resolution **student::student().**

Types of Constructors:-

1. **Default Constructors:** Default constructor is the constructor which doesn't take any argument. It has no parameters. i**t is also called a zero-argument constructor.**

```
class construct {
public:
        int a, b;
        construct() // Default Constructor
        {
                a = 10;
                b = 20;
        }
};

int main()
{
        construct c;
        cout << "a: " << c.a << endl << "b: " << c.b;
        return 1;
}
a: 10
b: 20
```

2. **Parameterized Constructors:** It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.

Note:- when the parameterized constructor is defined and **no default constructor is defined explicitly**, the compiler will not implicitly call the default constructor and hence creating a simple object as

Student s;

Will flash an error

```
class Point {
private:
	int x, y;

public:
	// Parameterized Constructor
	Point(int x1, int y1)
	{
		x = x1;
		y = y1;
	}

	int getX() { return x; }
	int getY() { return y; }
};

int main()
{
	// Constructor called
	Point p1(10, 15);

	// Access values assigned by constructor
	cout << "p1.x = " << p1.getX()
		<< ", p1.y = " << p1.getY();

	return 0;
}
```

p1.x = 10, p1.y = 15

. 3. **copy constructor:-** a constructor that is used to copy or initialize the value of one object into other object of same class is called copy constructor. Copy constructor takes a **reference to an object of the same class** as an argument.

```
 Sample(Sample &t) {
            id=t.id;
      }
```

Example: **Explicit copy constructor**

```
class Sample
{
      int id;
      public:
      void init(int x)
      {
            id=x;
      }
      Sample(){} //default constructor with empty body

      Sample(Sample &t) //copy constructor
      {
            id=t.id;
      }
      void display()
      {
            cout<<endl<<"ID="<<id;
      }
};
int main()
```

```
{
        Sample obj1;
        obj1.init(10);
        obj1.display();

        Sample obj2(obj1); //or obj2=obj1; copy constructor called
        obj2.display();
        return 0;
}
ID=10
ID=10
// Example: Implicit copy constructor
class Sample
{
        int id;
        public:
        void init(int x)
        {
                id=x;
        }
        void display()
        {
                cout<<endl<<"ID="<<id;
        }
};
```

```
int main()
{
        Sample obj1;
        obj1.init(10);
        obj1.display();
        Sample obj2(obj1); //or obj2=obj1; compiler create constructor by own
        obj2.display();
        return 0;
}
```

```
ID=10
ID=10
```

Qus 2 :- What is destructor?

Ans:- **Destructor:-** is an instance member function of a class

- The name of the destructor is same as the name of the class but preceded by **tilde(~)** symbol
- It can never be static.
- It has no return type.
- It take no argument (No overloading is possible).
- It is invoked implicitly when object is going to destroy.It is last function to run but do not destro object.
- ****It should be defined to release resources allocated to an object ***
- (object[pointer]--) -----> resource. **Exapmle**

```
Class A{
 int *p;
 A(){
    P = new int;
    *p  = 5;
}
```

```
~A(){
Delete p; //for free p memoery otherwise memory leak
}};
```

Example of destructor

```
class Complex{
   private:
      int a,b;
   public:
     ~Complex(){
          cout<<destructor;
      }
 };
void fun(){
Complex obj;
}
void main(){
 clrscr();
 fun(); // print destructor before getch
 getch();
}
```

Qus 3:- **What is the order of constructor and destructor execution in C++?**

Ans:- First base class constructor is executed and then derived class constructor, so execution happens from top to bottom in inheritance tree.**The order of destructor is exact opposite of constructor.**

Qus 4:- **What is initializer?**

- **Initializer List** is used to initialize data member of class.

- The list of members to be initialized is indicated as comma separated list followed by a colon.
- There are situations where initialization of data members inside constructor doesn't work(like refrence variable , const variable they require declaration with initialization both simultaneously) and Initializer list must be used.

1. For initialization of **non-static const data members**.
2. For initialization of **refrence variable.**

1.For initialization of non-static const data members:

```
1class Test {
   const int t;
    public:
    Test():t(5) {  }
     int getT() { return t; }
    };
 2 class Test {
     const int t;
  public:
     Test(int t):t(t) {}  //Initializer list must be used
     int getT() { return t; }};
```

2) For initialization of reference members:-

```
class Test {
   int &t;
   cons int x;
public:
   Test(int &t) : t(t) , x(5) {}  //Initializer list must be used
   int getT() { return t; }
};
Void main(){ int m = 5; Test t(m);
```

Qus 4). Explain constructor in inheritance?

Ans:

```
Class A{
  Public:
  A(){ }
};
Class B : public A{
  Public:
  B(){ }
};
Void main(){
  B obj;
```

```
Class A{
  Public:
  A(){ }
};
Class B : public A{
  Public:
  B() : A(){ }
};//created by compiler
Void main(){
  B obj;
```

Constructor **calling order**: B -> A.

Constructor **execution order** : A -> B.

Apposite for destructor

1.Yaha compiler default constructor dhudega derived class se.

```
Class A{
  int a;
  Public:
  A(int k){ a=k; }
};
Class B : public A{
  Public:
    B ( ) { }    error: no matching function for call to 'A::A()'
   // soln B():A(5){}
};// by compiler – compiler only create for default
Void main(){
  B obj;
```

```
}
2. class A{
int a;
  public:
  A(int k){ a =k; }
};
class B : public A{
  int b;
  public:
  B(int x , int y) : A(x) { b=x; }
};// by compiler
int main(){
  B obj(2,3);
  return 0;
}
```

Qus 5:- What is Dynamic constructor?

Ans:- When allocation of memory is done dynamically using dynamic memory allocator **new in a constructor**, it is known as dynamic constructor.

Construcor can allocate dynamically created memory to the object. Thus, object is going to use memory region which is dynamically created by constructor.

Example:-

```
Class A{
 int *p;
 A(){
    P = new int; // dynamic constructor  becz creating dynamic memory inside its body
    *p  = 5;
```

} };

Myth :- A *p = new A(); // **this is not dynamic constructor**

Qus6 :- What is virtual destructor?

Ans:-

when we use base class pointer or reference to hold the derived class object. If we don't have virtual destructor, then it will end up in **calling only base class destructor**.

```
class A{
  int *p;
  public:
  void f1(){}
   ~A(){delete p;}  //~A(){}
};
class B : public A{
  int *q;
  public:
  void f2(){}
  ~B(){delete q; }
};
int main(){
  //B* b = new A(); invalid conversion from 'A*' to 'B*'
  A* a = new B();
  a->f1();// early binding
  //a->f2(); error late binding 'class A' has no member named 'f2'
```

Delete a; // ye bas a class k member ko free karega so put virtual keyword before constructor **only p ki memory free hogi q ki nhi**

```
  return 0;
```

```
}
```