1 **What is constructor? List out characteristics of constructors.**

- A constructor is a “special” member function which initializes the objects of class.

Properties of constructor:

- Constructor is invoked automatically whenever an object of class is created.
- Constructor name must be same as class name.
- Constructors should be declared in the public section because private constructor cannot be invoked from outside the class so they are useless.
- Constructors do not have return types and they cannot return values, not even void.
- Constructors cannot be inherited, even though a derived class can call the base class constructor.
- Constructors cannot be virtual.
- An object with a constructor cannot be used as a member of a union.
- They make implicit calls to the operators **new** and **delete** when memory allocation is required.

Example :

```
class construct
 {
        public:  int a, b;          // Default Constructor
         construct()
         {
         a = 10;
         b = 20;
        }
};
int main()
{
 // Default constructor called automatically when the object is created
  construct c;
cout << "a: " << c.a << endl  << "b: " << c.b;
}
```

Output:

```
a: 10
b: 20
```

2 **Explain types of constructor with example.**

- There are mainly three types of constructors as follows:

1. Default constructor:

- Default constructor is the one which invokes by default when object of the class is created.
- It is generally used to initialize the value of the data members.
- It is also called no argument constructor.

Let's see the simple example of C++ default Constructor :

```
#include <iostream>
using namespace std;
class Employee
 {
  public:
     Employee()   // Default Constructor
```

```
    {
      cout<<"Default Constructor Invoked"<<endl;
    }
};
int main(void)
{
  Employee e1; //creating an object of Employee
  Employee e2;
  return 0;
}
```

Output :

Default Constructor Invoked
Default Constructor Invoked

2. Parameterized constructor :

- Constructors that can take arguments are called parameterized constructors.
- Sometimes it is necessary to initialize the various data elements of different objects with different values when they are created.
- We can achieve this objective by passing arguments to the constructor function when the objects are created.

Let's see the simple example of C++ Parameterized Constructor.

```
#include <iostream>
using namespace std;
    class Employee
     {
        public:
            int id;             //data member (also instance variable)
            string name;     //data member(also instance variable)
             float salary;
    Employee(int i, string n, float s)
    {
        id = i;
        name = n;
        salary = s;
    }
    void display()
     {
        cout<<id<<"  "<<name<<"  "<<salary<<endl;
    }
};
    int main()
    {
        Employee e1 =Employee(101, "abc", 890000);        //creating an object of Employee
        Employee e2=Employee(102, "xyz", 59000);
        e1.display();
        e2.display();
        return 0;
    }
```

Output :

```
101  abc  890000
102  xyz  59000
```

3. Copy constructor :

- A copy constructor is used to declare and initialize an object from another object.
- For example, integer(integer &i); OR integer I2(I1);
- Constructor which accepts a reference to its own class as a parameter is called copy constructor
- A copy constructor is a member function which initializes an object using another object of the same class.

Syntax :

Class_name(const class_name &old_object);

Let's see a simple example of the copy constructor. :

```
#include <iostream>
using namespace std;
    class A
    {
            public:
                    int x;
            A(int a)            // parameterized constructor.
            {
                    x=a;
            }
            A(A &i)             // copy constructor
            {
                    x = i.x;
            }
    };
    int main()
    {
            A a1(20);           // Calling the parameterized constructor.
            A a2(a1);           // Calling the copy constructor.
            cout<<a2.x;
            return 0;
    }
```

Output :

20

Example :

```
    #include<iostream>
    using namespace std;
    class rectangle
    {
            int length, width;
            public:
            rectangle() // Default constructor
            {
```

```
                length=0;
                width=0;
        }
        rectangle(int _length, int _width) // Parameterized constructor
        {
                length = _length;
                width = _width;
        }
        rectangle(rectangle &_r) // Copy constructor
        {
        length = _r.length;
        width = _r.width;
        }
                // other functions for reading, writing and processing can be written here
        };
int main()
{
        rectangle r1;            // Invokes default constructor
        rectangle r2(10,20);     // Invokes parameterized constructor
        rectangle r3(r2);        // Invokes copy constructor
}
```

Output :
1020
1020

3 What is destructor? Explain with example. List special properties of destructor.

- Destructor is used to destroy the objects that have been created by a constructor.
- Destructor is a member function whose name must be same as class name but is preceded by a tilde (~).
- Destructor never takes any argument nor it returns any value nor it has return type.
- Destructor is invoked automatically by the complier upon exit from the program.
- Destructor should be declared in the public section
- Example:

```
#include <iostream.h>
#include <conio.h>
class sample
{
    public:
    sample()   //Constructor
    {
            cout<<"Constructor\n";
    }
    ~sample()   //destructor
    {
            cout<<"Destructor\n";
    }
};
int main()
```

```
{
	cout<<"Main block\n";
	sample s1;
	{
		cout<<"Inner block\n";
		sample s2;
	}
	cout<<"Return back to main block\n";
	return 0;
}
```

OUTPUT :

Main block
Constructor
Inner block
Constructor
Destructor
Return back to main block
Destructor

4 **Explain Multiple Constructor in a Class(Constructor Overloading):**

- Still we had discussed two kind of constructors;
  1. No argument Constructor
  2. One argument Constructor
- In the first case, the constructor itself supplies the data values and no values are passed by the calling program.
- In the second case, the function call passes the appropriate values from main().
- C++ Permits us to use the both these constructors in the same class.
- **Example is shown below:**

```
#include<iostream>
using namespace std;
class example
{
	public:
	example()
	{
		cout<<"This is the No parameter Constructor Example"<<endl;
	}
	example(int a)
	{
		cout<<"This is the One parameter Constructor Example"<<endl;
	}
	example(int a,int b)
	{
		cout<<"This is the Two parameter Constructor Example"<<endl;
	}
};
int main()
{
```

```
        example e1;
        example e2(10);
        example e3(10,20);
        return 0;
}
```

Output:

This is the No parameter Constructor Example
This is the One parameter Constructor Example
This is the Two parameter Constructor Example

- It declares three constructors for an example object. The first constructor receives no arguments, the second constructor receives one argument and the third constructor will receive two arguments.
- If we had written example e1 then it will invoke the first constructor, another statement example e2(10) will invoke the second constructor with one argument, the last statement example e3(10,20) will invoke the third constructor with two arguments.

5 **Explain Constructor with default arguments**

- It is possible to define constructors with default arguments, For example, the constructor student can be declared as follows;
  student(int a,int b=9)
- The default value of the argument b is 9, Then the statement,
  student st(5);
- Assigns the values 5 to the 'a' variable and 0 to b. However if the statement;
  student st(5,10);
- Assigns 5 to 'a' variable and 10 to 'b' variable. The actual parameter, when specified, overrides the default value. As pointer out earlier, the missing arguments must be trailing once.
- It is important to understand the different between default constructor and default argument constructor.
- The default argument constructor can be called with either one arguments or no arguments. And when it is called with no arguments, it becomes a default constructor
- **Example is shown below:**

```
#include<iostream>
using namespace std;
class student
{
        public:
        student(int a,int b=9)
        {
                cout<<"This is the example of Default Argument Constructor"<<endl;
                int c=a+b;
                cout<<c;
        }
};
int main()
{
        student st(5);
        return 0;
}
```

Output:
This is the example of Default Argument Constructor
14

6 Explain dynamic constructor.

- Constructors are used to allocate memory while creating objects.
- We can also allocate memory at run time for object by using new operator in constructor is known as dynamic constructor.
- It is better solution to decrease the wastage of memory.

For example:

```
#include <iostream>
using namespace std;
class demo {
   const char* p;
public:
   // default constructor
   demo()
   {
      // allocating memory at run time
      p = new char[6];
      p = "demo";
   }

   void display()
   {
      cout << p << endl;
   }
};
int main()
{
   demo obj;
   obj.display();
}
```

OUTPUT :
demo