Constructors:

Constructors are member functions of a class which are used to initialize the data members of the class objects. These functions are automatically called when an object of its class is created. There is no need to call these functions.

Characteristics of constructors:

- The name of a constructor is same as that of class in which it is declared.
- Constructors do not have any return type, not even void.
- Constructors are always defined in the public section of the class.
- They cannot be inherited, though a derived class can call the base class constructor.
- A constructor may not be static.
- Like other C++ functions, constructors can also have default arguments.
- It is not possible to take the address of a constructor.
- Member functions may be called
- Constructors are not called directly
- Constructors show polymorphism in a class

Types of Constructors:

There are three types of constructors.
1. Default Constructors
2. Parameterized Constructors
3. Copy Constructors

1. Default Constructors

A constructor that accepts no parameter is called the default constructors.
Example

```
class stud
{
int sno; char sname[40];
public :
stud( ) // Default Constructor
{
sno=0;
strcpy(sname,"new" );
}
void getinfo( )
{ cin>> sno;
gets(sname);
}
void showinfo( )
{ cout<< sno;
puts(sname);
}
};
```

The above class stud has sno and sname as data members and three member functions i.e. stud (), getinfo(), showinfo( )

```
void main()
{
stud obj; // Default constructor called
Obj.showinfo(); // displays the value of sno as 0 and sname as "new"
Obj.getinfo(); // reads the user given value from the user
Obj.showinfo(); // displays the user given values
}
```

The default constructors are very useful when we want to create objects without having to type the initial values.

With a default constructor objects are created just the same way as variables of other data types are created.

2. Parameterized Constructors

A constructor that accepts parameters for its invocation is known as parameterized constructors.

Example

```
class stud
{
int sno; char sname[40];
Public :
stud(int s, char n[ ] ) // Paramerized Constructor
{
sno=s;
strcpy(sname,n);
}
void getinfo( )
{ cin>> sno;
gets(sname);
}
void showinfo( )
{ cout<< sno;
puts(sname);
}
};
```

This means we always specify the arguments whenever we declare an instance (object) of the class.

```
void main()
{
stud obj (1, "Ashu"); // Parameterized constructor invoked
Obj.showinfo(); // displays the value of sno as 1 and sname as "Ashu"
Obj.getinfo(); // reads the user given value from the user
Obj.showinfo(); // displays the user given values
}
```

Just like any other function a parameterized constructor can also have default arguments

```
stud(int s=0, char n[ ]="new\0" )
```

➢ A constructor with default arguments is equivalent to a default constructor.

➢·A class must not have a default arguments constructor and default constructor together as it generates ambiguity.

3. Copy Constructors:

A copy constructor is a constructor of the form classname(& classname). It is used to initialize an object with the values of another object.
The compiler will use the copy constructor whenever -
➢·We initialize an instance using values of another instance of same type.
➢·A function returns an object
➢·A function receives an object as parameter.

```
class stud
{
int sno; char sname[40];
Public :
stud( stud &s) // Copy Constructor
{
sno=s.sno;
strcpy(sname,s.name);
}
stud( ) // Default Constructor
{
sno=0;
strcpy(sname,"new" );
}
void getinfo( )
{ cin>> sno;
gets(sname);
}
void showinfo( )
{ cout<< sno;
puts(sname);
}
};
void main()
{
stud obj; // default constructor used
stud obj1(23, "Nisha"); // parameterized constructor used
stud obj2 = obj; // copy constructor used
stud obj3= obj1; // copy constructor used
```

Points to remember:
➢·Declaring a constructor with arguments hides the default constructor.
➢·A Constructor with default arguments is equivalent to a default constructor.

➢·A constructor declared under private access specifier, makes the class private and object of a private class cannot be created.

- A class must not have a default arguments constructor and default constructor together as it generates ambiguity.
- Constructors also show the polymorphism as a single class can have multiple constructors of different forms. (Also known as constructor / function overloading.)

1 Differentiate between default & copy constructor with a suitable example.

Ans: Default Constructor: A default constructor is a constructor that has no parameters.
Copy Constructor: A copy constructor is a constructor that that initialises a new object with the value of an existing object.

```
class abc
{
int x,y;
public:
abc( )
{
x = 10;
y = 20;
}
abc(abc &t)
{
x = t.x;
y = t.y;
}
};
```

13 What is a default constructor? How does it differ from destructor?

Ans: Default Constructor: A default constructor is a constructor that has no parameters.
Destructor: A destructor is used to released the resources and the memory allocated to the data member when scope of the object terminates

14 Explain the concept constructor overloading with a suitable example.

Ans: When multiple constructors are defined for a single class such constructors are referred to as overloaded constructor and the process is called constructor overloading.

```
class stud
{
int sno; char sname[40];
Public :
stud( stud &s) // Copy Constructor
{
sno=s.sno;
strcpy(sname,s.name);
}
stud( ) // Default Constructor
{
sno=0;
strcpy(sname,"new" );
}
void getinfo( )
{ cin>> sno;
gets(sname);
}
void showinfo( )
```

```
{ cout<< sno;
puts(sname);
}
};
```

Destructors:

1. Why is a destructor function required in classes? Illustrate with the help of an example.

A destructor is a class member function that has the same name as the constructor (and the class) but with a ~ (tilde) in front.
~stud() ;
Destructor is used to deinitialize or destroy the class objects. When an object goes out of scope, its destructor is automatically invoked to destroy the object.
Example :

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
class stud
{ int sno;
char sname[40];
public :
stud( ) // Default Constructor
{
sno=0;
strcpy(sname,"new");
cout<<"\nConstructing the object.....";
}
~stud( ) // Destructor
{
cout<< "\nDestructing the object.............";
}
void getinfo( )
{cout<<"\nEnter Student No. :";
cin>> sno;
cout<<"\nEnter Student Name :";
gets(sname);
}
void showinfo( )
{cout<< "\n Student No.: "<<sno;
cout<<"\n Student Name :";
puts(sname);
}
};
void main()
{clrscr();
stud obj; // Default constructor called
obj.showinfo(); // displays the value of sno as 0 and sname as "new"
obj.getinfo(); // reads the user given value for object obj
obj.showinfo(); // displays the user given values for object obj
getch();
```

```
}
```

2. What is the use of a constructor function in a class? Give a suitable example for a constructor function in a class.
   Ans: A constructor is a special member function of a class that has the same name as that of the class and is automatically called when an object of the class is declared. It allows the object a class to be initialized in the same way as variables or built- in data types.
   Example:

```
#include<iostream.h>
class abc
{
private:
int x;
public:
abc( ) //constructor function
{
x=10;
}
};
void main( )
{
abc a1;
}
```

ASSIGNMENT

Answer the questions (i) and (ii) after going through the following class:

```
class Exam
{ int Marks; char Subject[20];
public:
Exam () //Function 1
{ Marks = 0;
 strcpy (Subject,"Computer");
}
Exam(char S[]) //Function 2
{ Marks = 0;
strcpy(Subject,S);
 }
Exam(int M) //Function 3
{ Marks = M;
strcpy(Subject,"Computer");
 }
Exam(char S[], int M) //Function 4
{ Marks = M;
strcpy (Subject,S);
}
Exam(Exam &E); //Function 5
~Exam() //Function 6
{}
```

```
};
```

(i) Write statements in C++ that would execute Function 3 and Function 4 of class Exam.

(ii) Which feature of Object Oriented Programming is demonstrated using Function 1, Function 2, Function 3 and Function 4 in the above class Exam?

(iii) In Object Oriented Programming, what is Function 6 referred as and when does it get invoked/called?

(iv) Which category of constructor - Function 5 belongs to? Write complete definition of it.