Unit 3

Question Bank Solution

Q)Explain properties of constructor function with suitable program

- They will have the same name as of their class.
- They should be declared in the public section.
- They are executed automatically when objects are created.
- They do not have return types, not even void and thus they can't return values.
- They can't be inherited, though a derived class can call the base class constructor.
- They can have default arguments.
- Constructors can not be virtual.
- Their addresses can not be referred.
- An object with a constructor can not be used as a member of a union.
- They make 'implicit calls' to the operators new and delete when memory allocation is required

Example:-

```
#include<iostream>

Using namespace std;

Class Example {

 Int a, b; Public:

 Example() {

A = 10;

B = 20;

       Cout << "Im Constructor\n";

   }

  Void Display() {

       Cout << "Values :" << a << "\n" << b;

   }

};

Int main() {
   Example Object;
   Object.Display();
    Return 0;
}
```

Q)Design and write a suitable program to explain constructor overloading

```
#include <iostream>
Using namespace std;
Class construct
{
Public:
    Float area;
    Construct()
    {
        Area = 0;
    }
    Construct(int a, int b)
    {
        Area = a * b;
    }
Void disp()
{
     Cout<< area;
}
};
Int main()
{
```

```
Construct o;
  Construct o2( 10, 20);
    o.disp();
   o2.disp();
return 1;
}
```

Q)Explain merits of constructor and destructors.

Constructor:-

- Automatic initialization of objects at the time of their declaration.
- Multiple ways to initialize objects according to the number of arguments passes while declaration.
- The objects of child class can be initialized by the constructors of base class.

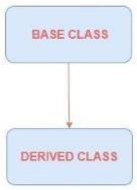
Destructor:-

- Destructors is called when lifetime of a program end.
- It is useful to make a final set of instructions.
- It gives the final chance to clean up the resources that are not in use to release the memory occupied by unused objects like deleting dynamic objects, close of the system handles, used files.
- Because of a lot of resources occupying space and not getting used in the computer, the destructor always comes with a good image to reduce the chances of memory leak by destroying those unused things.
- Though C++ does have the mechanism of Garbage Collection but calling the destructor automatically whether the programmer calls it or not to free up space prevents the user from many worst situations in the future.

Q)Explain visibility of inherited public, private and protected members using suitable C++ program.

```
#include <iostream>
Using namespace std;
Class Base {
  Private:
Int pvt = 1;
Protected:
Int prot = 2;
Public:
    Int pub = 3;
    Int getpvt() {
      Return pvt;
    }
};
Class Derived : public Base {
  Public:
    Int getProt() {
      Return prot;
    }
};
Int main() {
  Derived object1;
  Cout << "Private = " << object1.getpvt();
  Cout << "Protected = " << object1.getProt();
  Cout << "Public = " << object1.pub;
  Return 0;
}
```

Q) Write a C++ program to describe single inheritance Single Inheritance:- As shown in the above diagram, in C++ single inheritance, a single class can be derived from the base class. When we create a derived class from a base class, we use access specifiers to inherit the properties of the base class. Access specifiers can be private, protected, or public.



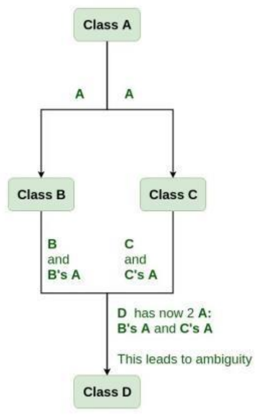
Program:-

```
#include<iostream> using

namespace std; class

father

{

public:

   void house()

   {

      cout<<"Have own 2BHK House.";

   } }; class

son:public father

{    public:

void car()

   {

      cout<<"Have own Audi Car.";
```

```
    } }; int

main() {

son obj;

obj.house();

obj.car();

return 0;

}
```

Q) Describe virtual base class with example.

Virtual base class:- When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word virtual.



Program:-

```
#include <iostream>

Using namespace std;
```

```
Class     A     {

Public:

   Int a;

   A(){

      A = 10;

   }

};

Class B : public virtual A { };

Class C : public virtual A { };

Class D : public B, public C { };

Int main(){

  D object;

   Cout << “a = “ << object.a << endl;

   Return 0;

}
```

Q) Explain concept of abstract class with suitable C++ program

Abstract class:- An abstract class in C++ is a class that contains at least one pure virtual function, and these classes cannot be instantiated. Abstract Classes came from the idea of abstraction. Before we dive into technicalities, let’s first consider a scenario where you might use this concept.

Program:-

```
#include <iostream>
```

```
Using namespace std;

 Class Test {

    Int x;

Public:

Virtual void show() = 0;

  Int getX() { return x; }

};

Int main()

{

 Test t;

Return 0;

}
```