**Government Polytechnic College, Jodhpur**
**Department of Computer Science** (NBA Accredited)

**Programme**: Diploma **Class Test**: II **Term**: 2017-18
**Course**: Object Oriented Programming through C++ **Year**: IIIrd
**Course CODE:** CS-302 **Time:**04:00 to 05:00
**Max.Marks :**15 **Date:** 22-01-2018
**Instructions to candidates:** Attempt Any Three Questions
1.Mobiles,smart watches or any electronic gadgets are strictly banned.

| SI# | Question | Marks | CO MAPPING |
|---|---|---|---|
| 1 | What is Constructor? Explain the types of Constructors with examples. | 5 | CO3 |
| 2 | What is friend function? Explain with programming example. | 5 | CO3 |
| 3 | What is Function overloading? Explain with programming example. | 5 | CO4 |
| 4. | Write a program to add two complex no. using operator overloading | 5 | CO4 |

## 1. WHAT IS CONSTRUCTOR? EXPLAIN THE TYPES OF CONSTRUCTORS WITH EXAMPLES.

A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object (instance of class) create. It is special member function of the class.

A constructor is different from normal functions in following ways:

- Constructor has same name as the class itself
- Constructors don't have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

**Types of Constructors**

1. **Default Constructors:** Default constructor is the constructor which doesn't take any argument. It has no parameters. Even if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly. The default value of variables is 0 in case of automatic initialization.
2. **Parameterized Constructors:** It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object. The constructors can be called explicitly or implicitly.
3. **Copy constructor**: A copy constructor is a member function which initializes an object using another object of the same class. It is a special type of constructor which takes an object as an argument and is used to copy values of data members of one object into another object. The copy constructors are used to declaring and initializing an object from another object. If we don't define our own copy constructor, the C++ compiler creates a default copy constructor for each class which does a member wise copy between objects.

```
class Complex
{
 private:
    int R,I;
 public:
   void Read();
   void Display();
   Complex()
   {
    cout<<"\n Default Constructor Called";
    R=0;
    I=0;
   }
   Complex(int r, int i)
   {
```

```
    cout<<"\n Parameterized Constructor Called";
    R=r;
    I=i;
    }
   Complex( Complex & C)
    {
    cout<<"\n Copy Constructor Called";
    R=C.R;
    I=C.I;
    }
};
void main()
{
 Complex C1;
 Complex C2(10,20);
 Complex C3(C2);
 C1.Display();
 C2.Display();
 C3.Display();
}
```

**2. WHAT IS FRIEND FUNCTION? EXPLAIN WITH PROGRAMMING EXAMPLE.**

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword **friend** as follows –

**Rules of friend function:**

- It defined without accessing the scope of the class.
- It can work similar global function when the argument is not a class.
- It cannot use the member directly private or public.
- The private member can be changed or accessed only through its object argument.
- It can be defined as inline function outside the class only not in prototype.
- It can be a public or private function without any effect of its meaning.
- It usually has class as its arguments only then it can be used among defined class arguments.

```
#include<iostream.h>
#include<conio.h>
class Sample
{
        private:
                int a,b;
        public:
                void Read();
                friend void Sum(sample &);
                void Display();
};
void Sample :: Read();
{
        cout<<"enter the value of a:";
        cin>>a;
        cout<<"enter the value of b:"
        cin>>b;
}
void Sum(sample &x)
{
        x.s=x.a+x.b;
}
void Sample::Display();
{
        cout<<"sum"<<s;
}
void main()
{
```

```
        Clrscr();
        Sample s;
        s.Read();
        sum(s);
        s.Display();
        getch();
}
```

**3. WHAT IS FUNCTION OVERLOADING? EXPLAIN WITH PROGRAMMING EXAMPLE**

Function overloading is a feature in C++ where two or more functions can have the same name but different parameters. C++ allows you to specify more than one definition for a function name or an operator in the same scope, which is called function overloading and operator overloading respectively.

An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

We can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. You cannot overload function declarations that differ only by return type.

```
#include <iostream>
#include <conio.h>
#include <math.h>
#define pie 3.14

 float area(float a)
 {
  return (pie*a*a);
 }

 float area(float a, float b)
 {
  return (a*b);
 }

 float area(float a, float b, float c)
 {
  float s, temp;
  s = (a+b+c)/2;
  temp = s*(s-a)*(s-b)*(s-c);
  temp = pow(temp,0.5);
  return temp;
 }

void main()
{
  cout<<"Area of circle="<<area(10);
 cout<<"Area of rectangle=" << area(10,20);
 cout<<"Area of triangle ="<<area(5,5,5);
 getch();
}
```

**4. WRITE A PROGRAM TO ADD TWO COMPLEX NO. USING OPERATOR OVERLOADING**

```
class Complex
{
 private:
    int R,I;
 Public:
   void Read()
   {
     cout<<"\nEnter the Real and Imag. part: ";
     cin>>R>>I;
   }
   void Display()
```

```
    {
     cout<<R<<"+i"<<I;
    }
   Complex operator+(Complex C2)
   {
     Complex temp;
       temp.R=R + C2.R;
       temp.I= I + C2.I;
    return temp;
  }
};

void main()
{
 Complex C1,C2,C3;
 C1.Read();
 C2.Read();
 C3=C1+C2;
 C3.Display();
}
```