DATA STRUCTURES LABORATORY MANUAL – ICE 2144

III SEMESTER B. TECH

EXPT. 4 OPERATOR OVERLOADING

1. Write a program to demonstrate the i) Operator Overloading ii) Function Overloading.

i) Operator Overloading: -The mechanism of giving a special meaning to an operator is called operator overloading. This can be achieved by special function "operator"

Syntax:

```
return type classname:: operaotor op(list of arguments)
{
        …….
}
```

Program:

```
#include<iostream>
using namespace std;

class complex
{
        float real, img;
        public:
                complex();
                complex(float x, float y);
                void read_complex();
                complex operator+(complex);
                complex operator-(complex);
                void display();
};
complex::complex()
{
        real=img=0;
}
complex::complex(float x, float y)
{
        real=x;
        img=y;
}
void complex::display()
{
        char sign;
        if(img<0)
        {
                sign='-';
                img = -img;

        }
```

```
        else
        {
                sign='+';
         }
        cout<<real<<sign<<"i"<<img<<endl;
}
complex complex::operator+(complex c)
{
        complex r;
        r.real=real+c.real;
        r.img=img+c.img;
        return r;
}
complex complex::operator-(complex c)
{
        complex r;
        r.real=real-c.real;
        r.img=img-c.img;
        return r;
}
void complex::read_complex()
{
        cout<<"Enter real part of complex number:";
        cin>>real;
        cout<<"Enter Imaginary part of complex number:";
        cin>>img;
}
int main()
{
        complex a;
        a.read_complex();
        complex b;
        b.read_complex();
        complex c;
        c=a+b;
        cout<<"After Addition of two complex numbers:";
        c.display();
        c=a-b;
        cout<<"Difference of two complex numbers:";
        c.display();
        return 0;
}
```

ii) Function Overloading

```
#include<iostream>
using namespace std;

class printData
{
        public:
                void print(int i)
                {
                        cout<<"Printing int: "<< i <<endl;
                }
                void print(double f)
                {
                        cout<<"Printing float: "<< f <<endl;
                }
                void print(char*c)
                {
                        cout<<"Printing string: "<< c <<endl;
                }
};
int main(void)
{
        printData pd;           // Call print to print integer
        pd.print(5);            // Call print to print float
        pd.print(500.263);      // Call print to print character
        pd.print("Welcome to Dept. of ICE");
        return 0;
}
```

2. Write a Program to demonstrate friend function and friend class.

```
#include<iostream>
using namespace std;

class sample2;
class sample1
{
        int x;
        public:
                sample1(int a);
                friend void max(sample1 s1,sample2 s2);
};
sample1::sample1(int a)
{
        x=a;
```

```
}
class sample2
{
        int y;
        public:
                sample2(int b);
                friend void max(sample1 s1,sample2 s2);
};
sample2::sample2(int b)
{
        y=b;
}
void max(sample1 s1,sample2 s2)
{
        if(s1.x>s2.y)
                cout<<"Data member in Object of class sample1 is larger "<<endl;
        else
                cout<<"Data member in Object of class sample2 is larger "<<endl;
}
int main()
{
        sample1 obj1(3);
        sample2 obj2(5);
        max(obj1, obj2);
        return 0;
}
```

EXERCISE:

1. Write a C++ program to perform various arithmetic operations on two complex numbers using friend functions.
2. Write a C++ program to find areas of different shapes (square, rectangle and triangle).
3. Write a C++ program that illustrates unary operator overloading for ++ and -- operators.
4. Write a C++ program for Binary operator overloading.
5. Write a C++ program for Member function Overloading with in the class.