Lab-9

Title: Polymorphism

Objective:

• To be familiar with concept of compile time and run time polymorphism

Theory:

• Introduction to polymorphism

• Types of polymorphism

i. Compile time polymorphism
   - Function overloading
   - Operator overloading

ii. Runtime polymorphism
   - Virtual function

1. Write a program finding area of square, rectangle, triangle by using function overloading technique.

```
#include<iostream>
using namespace std;
class Area{
    public:
    int square(int l){
        return(l*l);
    }
    int rectangle(int l,int b){
        return(l*b);
    }
    float triangle(float b,float h){
        return((b*h)/2);
    }
};
    int main(){
        Area a;
        cout<<"Area of square:"<<a.square(5)<<endl;
        cout<<"Area of rectangle:"<<a.rectangle(5,4)<<endl;
        cout<<"Area of triangle:"<<a.triangle(5,4)<<endl;
        return 0;
    }
```

2. Write a simple program to overload unary minus(-) operator.

```
#include<iostream>
using namespace std;
class Minus{
    private:
        int x;
        int y;
        int z;
    public:
        void set_data(int a,int b,int c){
                x=a;
                y=b;
                z=c;
        }
        void display(){
                cout<<"x="<<x<<endl;
                cout<<"y="<<y<<endl;
                cout<<"z="<<z<<endl;
      }
        void operator-(){
                x=-x;
                y=-y;
                z=-z;
        }
};
int main(){
    Minus m;
    m.set_data(5,6,7);
    cout<<"m:"<<endl;
    m.display();
    -m;
    cout<<"-m:"<<endl;
    m.display();
    return 0;

}
```

3. Write a program to generate Fibonacci series using operator overloading of ++operator

```
#include<iostream>
using namespace std;
class Fibo{
        private:
                int a,b,c;
        public:
                Fibo(){
                        a=-1;
                        b=1;
                        c=a+b;
                }
                void display(){
                        cout<<c<<",";

                }
                void operator++(){
                        a=b;
                        b=c;
                        c=a+b;
                }
};
int main(){
        int n,i;
        Fibo f;
        cout<<"Enter the number of term"<<endl;
        cin>>n;
        for(i=0;i<n;i++){
                f.display();
                ++f;
        }
        return 0;

}
```

4. WAP to add two complex numbers. Your program should have three objects. Each object contains two attributes (i.e. real and imaginary part) Now add each attribute and save them into third object separately. Use the concept of '+' operator overloading.

```
#include<iostream>
using namespace std;
class Complex{
        private:
                int real,imag;
        public:
        Complex()
        {
        }
        Complex(int r,int i){
                real=r;
                imag=i;
        }
        void display(){
                cout<<real<<"+i"<<imag<<endl;
        }
        Complex operator +(Complex c2){
                Complex temp;
                temp.real=real+c2.real;
                temp.imag=imag+c2.imag;
                return temp;
        }
};
int main(){
        Complex c1(4,6);
        Complex c2(4,6);
        Complex c3;
        cout<<"c1:"<<endl;
        c1.display();
        cout<<"c2:"<<endl;
        c2.display();
        c3=c1+c2;
        cout<<"Sum of Complex"<<endl;
        c3.display();
        return 0;
}
```

5. Write a Program to find sum of two 3 X 3 matrix by overloading binary operator +.

```
#include<iostream>
using namespace std;
class Matrix{
        private:
                int M[3][3];
        public:
                void get_matrix(){
                        cout<<"Enter the 3X3 matrix:"<<endl;
                        for(int i=0;i<3;i++){
                                for(int j=0;j<3;j++){
                                        cin>>M[i][j];
                                }
                        }
                }
                void display_matrix(){
                        cout<<"Entered the 3X3 matrix:"<<endl;
                        for(int i=0;i<3;i++){
                                for(int j=0;j<3;j++){
                                        cout<<M[i][j]<<" ";
                                }
                                cout<<endl;
                        }
                }
                Matrix operator +(Matrix m){
                        Matrix R;
                        for(int i=0;i<3;i++){
                                for(int j=0;j<3;j++){
                                        R.M[i][j]=m.M[i][j]+M[i][j];
                                }
                        }
                        return R;
                }
};
int main(){
        Matrix A,B,C;
        cout<<"Enter the element for matrix A"<<endl;
        A.get_matrix();
        cout<<"Enter the element for matrix B"<<endl;
        B.get_matrix();
        C=A+B;
        C.display_matrix();
        return 0;
        }
```

6. Write a Program to overload multiplication operator(*) showing the multiplication of two objects.*/

```
#include<iostream>
using namespace std;
class Mult{
        int a,b;
        public:
        Mult(){
        }
        Mult(int m){
                a=m;
        }
        Mult operator *(Mult m2){
                Mult temp;
                temp.a=m2.a*a;
                return temp;
        }
        void display(){
                cout<<"Multiplication of two object:"<<a<<endl;
        }
};
int main(){
        Mult m1(4);
        Mult m2(5);
        Mult m3;
        m3=m1*m2;
        m3.display();
        return 0;
}
```

7. Write a program to implement vector addition using operator overloading
    i. Using Friend Function

```
#include<iostream>
using namespace std;
class Vector{
        private:
                int x,y,z;
        public:
        Vector(){

        }
        Vector(int a,int b,int c){
                x=a;
                y=b;
                z=c;
        }
        void display(){
                cout<<x<<"i+"<<y<<"j+"<<z<<"j"<<endl;
        }
        friend Vector operator +(Vector v1,Vector v2);
};
Vector operator +(Vector v1,Vector v2){
        Vector temp;
        temp.x=v1.x+v2.x;
        temp.y=v1.y+v2.y;
        temp.z=v1.z+v2.z;
        return temp;
}
int main(){
        Vector v1(5,6,7);
        Vector v2(8,9,7);
        Vector v3;
        v3=v1+v2;
        cout<<"for vector v1:";
        v1.display();
        cout<<"for vector v2:";
        v2.display();
        cout<<"Sum of vector v1 and v2:";
        v3.display();
        return 0;
}
```

ii. Without using Friend Function (using member function)

```
#include<iostream>
using namespace std;
class Vector{
	private:
		int x,y,z;
	public:
	Vector(){

	}
	Vector(int a,int b,int c){
		x=a;
		y=b;
		z=c;
	}
	void display(){
		cout<<x<<"i+"<<y<<"j+"<<z<<"j"<<endl;
	}
Vector operator +(Vector v2){
	Vector temp;
	temp.x=x+v2.x;
	temp.y=y+v2.y;
	temp.z=z+v2.z;
	return temp;
}};
int main(){
	Vector v1(5,6,7);
	Vector v2(8,9,7);
	Vector v3;
	v3=v1+v2;
	cout<<"for vector v1:";
	v1.display();
	cout<<"for vector v2:";
	v2.display();
	cout<<"Sum of vector v1 and v2:";
	v3.display();
	return 0;
}
```

8. Create a base class student. Use the class to store name, dob, rollno and includes the member function getdata(), discount().Derive two classes PG and UG from student.. make dispresult() as virtual function in the derived class to suit the requirement.

```
#include<iostream>
using namespace std;
class Student
        {
        protected:
                char name[20],dob[10];
                int roll;
                float dis,fees;
        public:
        void getdata()
                        {
                        cout<<"Enter the name of student"<<endl;
                        cin>>name;
                        cout<<"Enter the roll number of student"<<endl;
                        cin>>roll;
                        cout<<"Enter the DOB in yy-mm-dd"<<endl;
                        cin>>dob;
                        cout<<"Enter the fees"<<endl;
                        cin>>fees;
                        }
          void discount()
                        {
                        cout<<"Enter discount given to student"<<endl;
                        cin>>dis;
                        }
        virtual void dispresult()
        { }
        };
        class UG:public Student{
        private:
                float paidamount;
        public:
          void dispresult()
                        {
                        paidamount=fees-dis;
                        cout<<"Name:"<<name<<endl;
                        cout<<"Roll no:"<<roll<<endl;
                        cout<<"Date of Birth:"<<dob<<endl;
                        cout<<"Discount:"<<dis<<endl;
                        cout<<"Fees:"<<fees<<endl;
                        cout<<"Fees to be paid="<<paidamount<<endl;
```

```
                    }
};
class PG:public Student
{
private:
   float paidamount;
public:
        void dispresult()
                {
                paidamount=fees-dis;
                cout<<"Name:"<<name<<endl;
                cout<<"Roll no:"<<roll<<endl;
                cout<<"Date of Birth:"<<dob<<endl;
                cout<<"Discount:"<<dis<<endl;
                cout<<"Fees:"<<fees<<endl;
                cout<<"Fees to be paid="<<paidamount<<endl;
                }
};
int main()
{
        Student *st1,*st2;
        PG p;
        UG u;
        st1=&p;
        st2=&u;
        cout<<"Enter PG student details:"<<endl;
        st1->getdata();
        st1->discount();
        cout<<"Enter UG student details:"<<endl;
        st2->getdata();
        st2->discount();
        cout<<"Details of PG student:"<<endl;
        st1->dispresult();
        cout<<"Details of UG student:"<<endl;
        st2->dispresult();
        return 0;
}
```

9. Create a abstract class shape with two members base and height, a member function for initialization and a pure virtual function to compute area().Derive two specific classes, Triangle and Rectangle which override the function area ().Use these classes in main function and display the area of triangle and rectangle.

```
#include<iostream>
using namespace std;
class Shape
{
protected:
      float base, height;
public:
     void setdimension(float x, float y)
          {
            base=x;
            height=y;
            }
virtual void area() =0;
} ;
class Triangle : public Shape
{
public:
     void area()
         {
          cout<<"Area of triangle="<<(0.5*base*height)<<endl;
          }
};
class Rectangle : public Shape
{
public:
    void area()
        {
         cout<<"Area of Rectangle="<<(base*height)<<endl;
         }
};
int main()
{
    Shape *bptr;
    Triangle t;
    Rectangle r;
    t.setdimension(10.0, 5.0);
    r.setdimension(20.0, 10.0);
    bptr= &t;
    bptr->area();
    bptr = &r;
```

```
    bptr->area();
    return 0;
}
```

10. We have a class name Complex which have data member real and imaginary. Default and parametrized constructor are there to initialize data member. Write a program to overloading unary operator '++' where real and imaginary data member will be incremented and '+' operator to add two complex number.

```
#include<iostream>
using namespace std;
class Complex
{
 int real, imag;
 public:
 Complex()
{
real=0;
imag=0;
}
 Complex(int r, int i)
 {
 real=r;
 imag=i;
 }
 Complex operator++()
 {
 real++;
 imag++;
 return *this;
 }
 Complex operator +(Complex c2)
 {
 Complex temp;
 temp.real = real + c2.real;
 temp.imag = imag + c2.imag;
 return temp;
 }
void display()
 {
 cout << real << "+" << imag <<"i"<<endl;
 }
};
int main()
{
```

```
 Complex c1(10, 5);
 Complex c2(2, 4);
 Complex c3;
 ++c1;
 ++c2;
 c3= c1 + c2;
 c3.display();
 return 0;
}
```