ASSIGNMENT NO:3 **DATE:04/07/2015**

PROGRAM TITLE:Write a Program to perform the following operations on fractions

1.e=-a+b*d

2.f=(c*d)/a

using operator overloading. Now check if e and f are equal.

THEORY: Fraction consists of two integers named denominator and numerator. The denominator cannot be zero in a fraction.

Operator overloading is a way by which we can extend upon the functionalities of normal operators so that they can be applied to objects as well.

PROGRAM ALGORITHM:

//a and b are the fractions where $a=\frac{p}{q}, b=\frac{x}{y}$

```
Algo_fraction_+(a,b)
{
```

$$\frac{p}{q}+\frac{x}{y}=\frac{p.y+x.q}{q.y}$$

```
}
Algo_fraction_-(a,b)
{
```

$$\frac{p}{q}-\frac{x}{y}=\frac{p.y-x.q}{q.y}$$

```
}
Algo_fraction_*(a,b)
{
```

$$\frac{p}{q}*\frac{x}{y}=\frac{p.x}{q.y}$$

```
}
Algo_fraction_/(a,b)
{
```

$$\frac{p}{q}/\frac{x}{y}=\frac{p.y}{q.x}$$

```
}
```

PROGRAM CODE:

```
/*C++ Program to implement functions on Fraction using operator
overload*/
#include<iostream>
using namespace std;

/*Class Fraction and its associated functions*/
class Fraction
{
     int p,q;
     public:
          Fraction();
          void input();
          void reduce();
```

```
            Fraction operator+(Fraction);
            Fraction operator-();
            Fraction operator-(Fraction);
            Fraction operator*(Fraction);
            Fraction operator/(Fraction);
            int operator==(Fraction);
            void display();
};

/*Non-Parameterised Constructor*/
Fraction::Fraction():p(0),q(1)
{
}

/*Function to take input*/
void Fraction::input()
{
      do
      {
            cout<<"\tEnter Denominator:";
            cin>>q;
      }
      while(q==0);
      cout<<"\tEnter Numerator:";
      cin>>p;
      if(q<0)
      {
            p=-p;
            q=-q;
      }
}

/*Fucntion to reduce the fraction*/
void Fraction::reduce()
{
      int i,gcd,flag=0;
      if(q<0)
      {
            p=-p;
            q=-q;
      }
      if(p<0)
      {
            flag=1;
            p=-p;
      }
      for(i=1;i<=(p*q);i++)
      {
            if((p%i==0)&&(q%i==0))
                  gcd=i;
      }
      p=p/gcd;
      q=q/gcd;
      if(flag)
            p=-p;
```

```
}

/*Overloading binary operator +*/
Fraction Fraction::operator+(Fraction a)
{
      Fraction b;
      b.p=p*a.q+a.p*q;
      b.q=q*a.q;
      return b;
}

/*Overloading unary operator -*/
Fraction Fraction::operator-()
{
      p=-p;
      return *this;
}

/*Overloading binary operator -*/
Fraction Fraction::operator-(Fraction a)
{
      Fraction b;
      b.p=p*a.q-a.p*q;
      b.q=q*a.q;
      return b;
}

/*Overloading binary operator **/
Fraction Fraction::operator*(Fraction a)
{
      Fraction b;
      b.p=p*a.p;
      b.q=q*a.q;
      return b;
}

/*Overloading binary operator /*/
Fraction Fraction::operator/(Fraction a)
{
      Fraction x;
      x.p=p*a.q;
      x.q=q*a.p;
      return x;
}

/*Overloading relational operator ==*/
int Fraction::operator==(Fraction a)
{
      if((a.p==p)&&(a.q==q))
            return 1;
      else
            return 0;
}

/*Displaying the fraction*/
```

```
void Fraction::display()
{
	reduce();
	if(q>1)
		cout<<p<<"/"<<q<<endl;
	else
		cout<<p<<endl;
}
int main()
{
	Fraction a,b,c,d,e,f;

	/*Creating the objects for performing operations*/
	cout<<"\n\tEnter fraction a="<<endl;
	a.input();
	cout<<"\n\tEnter fraction b="<<endl;
	b.input();
	cout<<"\n\tEnter fraction c="<<endl;
	c.input();
	cout<<"\n\tEnter fraction d="<<endl;
	d.input();
	cout<<"\n\ta=";
	a.display();
	cout<<"\n\tb=";
	b.display();
	cout<<"\n\tc=";
	c.display();
	cout<<"\n\td=";
	d.display();
	cout<<"\n\tExpression 1::\n\te=-a+b*d";
	e=-a+b*d;
	cout<<"\n\te=";
	e.display();
	cout<<"\n\tExpression 2::\n\tf=(c*d)/a";
	f=(c*d)/a;
	cout<<"\n\tf=";
	f.display();
	if(e==f)
		cout<<"\n\te is equal to f"<<endl;
	else
		cout<<"\n\te is not equal to f"<<endl;
	return 0;
}
```

OUTPUT:

```
	Enter fraction a=
	Enter Denominator:2
	Enter Numerator:3

	Enter fraction b=
	Enter Denominator:0
	Enter Denominator:-2
	Enter Numerator:9
```

```
Enter fraction c=
Enter Denominator:4
Enter Numerator:16

Enter fraction d=
Enter Denominator:-4
Enter Numerator:-9

a=3/2

b=-9/2

c=4

d=9/4

Expression 1::
e=-a+b*d
e=-93/8

Expression 2::
f=(c*d)/a
f=-6

e is not equal to f
```

DISCUSSION:

After applying the unary operation '-' on 'a', the value of 'a' gets changed for the remainder of the program.
The 'reduce' function uses gcd to represent the fraction into its simplest form.