Programming Assignment 2

Write a rational number class. This problem will be revisited in Chapter11, where operator overloading will make the problem much easier. For now we will use member functions add, sub, mul, div, and less that each carry out the operations +, -, *, /, and <. For example, x + y will be written x.add(y), and x < y will be written x.less(y).

Define a class for rational numbers. A rational number is a "rational" number, composed of two integers with division indicated. The division is not carried out, it is only indicated, as in 1/2, 2/3, 15/32, 65/4, 16/5. You should represent rational numbers by two int values, numerator and denominator.

A principle of abstract data type construction is that constructors must be present to create objects with any legal values. You should provide constructors to make objects out of pairs of int values; this is a constructor with two int parameters. Since every int is also a rational number, as in 2/1 or 17/1, you should provide a constructor with a single int parameter.

Provide member functions input and output that take an istream and ostream argument, respectively, and fetch or write rational numbers in the form 2/3 or 37/51 to or from the keyboard (and to or from a file).

Provide member functions add, sub, mul, and div that return a rational value. Provide a function less that returns a bool value. These functions should do the operation suggested by the name. Provide a member function neg that has no parameters and returns the negative of the calling object.

Provide a main function that thoroughly tests your class implementation. The following formulas will be useful in defining functions.

$a/b + c/d = (a * d + b * c) / (b * d)$

$a/b - c/d = (a * d - b * c) / (b * d)$

$(a/b) * (c/d) = (a * c) / (b * d)$

$(a/b) / (c/d) = (a * d) / (c * b)$

$-(a/b) = (-a/b)$

$(a/b) < (c/d)$ means $(a * d) < (c * b)$

$(a/b) == (c/d)$ means $(a * d) == (c * b)$

Let any sign be carried by the numerator; keep the denominator positive.

What to submit:

Create a directory named Prog2 in your home directory under 280.
Copy your ".h" and ".cpp" files to this directory.

```
//Main.cpp
// Note that I have overloaded functions for << and >> operations.

#include <iostream>
#include "rational.h"
using namespace std;
int main()
{
  cout << "Testing declarations" << endl;
  cout << "Rational x, y(2), z(-5,-6), w(1,-3);" << endl;
  Rational x, y(2), z(-5,-6), w(1,-3);
  cout << "x = " << x << ", y = " << y << ",  z = " << z
       << ", w = " << w << endl;

  cout << "Enter "
       << "a fraction in the format "
       << "integer_numerator/integer_denominator"
       << endl;
  cin >> x;
  cout << "You entered the equivalent of: " << x << endl;
  cout << z << " -  (" << w << ") = " << z - w << endl;

  cout << "Testing the constructor and normalization routines: " << endl;
  y =Rational(-128, -48);
  cout << "y =Rational(-128, -48) outputs as " << y << endl;
  y =Rational(-128, 48);
  cout << "y =Rational(-128, 48)outputs as " << y << endl;
  y = Rational(128,-48);
  cout << "y = Rational(128, -48) outputs as " << y << endl;
  Rational a(1,1);
  cout << "Rational a(1,1); a outputs as: " << a << endl;
  Rational ww = y*a;
  cout <<  y << " * " << a << " = " << ww << endl;

  w = Rational(25,9);
  z = Rational(3,5);
  cout << "Testing arithmetic and relational "
       << " operator overloading" << endl;
  cout << w << " mul " << z << " = " << mul(w , z) << endl;
  cout << w << " add " << z << " = " << add(w , z) << endl;
  cout << w << " sub " << z << " = " << sub(w , z) << endl;
  cout << w << " div " << z << " = " << div(w , z) << endl;

  cout << w << " less  " << z << " = " << less(w , z) << endl;
  cout << w << " less " << w << " = " << less(w , w) << endl;


  w = Rational(-21,9);
  z = Rational(3,5);
  cout << w << " mul " << z << " = " << mul(w , z) << endl;
  cout << w << " add " << z << " = " << add(w , z) << endl;
  cout << w << " sub " << z << " = " << sub(w , z) << endl;
  cout << w << " div " << z << " = " << div(w , z) << endl;
  cout << w << " less  " << z << " = " << less(w , z) << endl;
  cout << w << " less " << w << " = " << less(w , w) << endl;

  cout << neg(w) << " neg " << " = " << neg(neg(w)) << endl;
  cout << w << " neg " << " = " << neg(w) << endl;

  return 0;
}
```

Sample test Run

```
admin-147-64:Rational_numbers gurajas$ cat rational.out

Testing declarations

Rational x, y(2), z(-5,-6), w(1,-3);

x = 0/1, y = 2/1,  z = 5/6, w = -1/3


Enter a fraction in the format integer_numerator/integer_denominator

You entered the equivalent of: 9/7


Testing the constructor and normalization routines:

y =Rational(-128, -48) outputs as 8/3

y =Rational(-128, 48)outputs as -8/3

y =Rational(128, -48) outputs as -8/3

Rational a(1,1); a outputs as: 1/1

-8/3 mul 1/1 = -8/3

Testing arithmetic and relational operator overloading

25/9 mul 3/5 = 5/3

25/9 add 3/5 = 152/45

25/9 sub 3/5 = 98/45

25/9 div 3/5 = 125/27
```

```
25/9 less 3/5 = 0
25/9 less 25/9 = 0
-7/3 mul 3/5 = -7/5
-7/3 add 3/5 = -26/15
-7/3 sub 3/5 = -44/15
-7/3 div 3/5 = -35/9
-7/3 less 3/5 = 1
-7/3 less -7/3 = 0
7/3 neg = -7/3
-7/3 neg = -7/3
```