EEL3834 - Programming for Electrical Engineers
Fall 2016
Programming Assignment 8
Assigned: 11/11/2016 Due: 11/18/2016 @ 5:00PM
To be done individually

Programming Project 2 from Chapter 8 of Absolute C++ 5th ed. (Savitch), pg. 364:

Define a class for rational numbers. A rational number is a number that can be represented as the quotient of two integers. For example, 1/2, 3/4, 64/2, and so forth are all rational numbers. (By 1/2 and so on we mean the everyday fraction, not the integer division this expression would produce in a C++ program.) Represent rational numbers as two values of type int, one for the numerator and one for the denominator. Call the class Rational.

Include a constructor with two arguments that can be used to set the member variables of an object to any legitimate values. Also include a constructor that has only a single parameter of type int; call this single parameter wholeNumber and define the constructor so that the object will be initialized to the rational number wholeNumber /1. Include a default constructor that initializes an object to 0 (that is, to 0/1). You should include a function to normalize the values stored so that, after normalization, the denominator is positive and the numerator and denominator are as small as possible. For example, after normalization 4/-8 would be represented the same as -1/2. To help with the normalize function, it may be helpful to also implement a greatest common divisor function.

Overload the input and output operators >> and << . Numbers are to be input and output in the form 1/2, 15/32, 300/401, and so forth. Note that the numerator, the denominator, or both may contain a minus sign, so -1/2, 15/-32, and -300/-401 are also possible inputs. Overload all the following operators so that they correctly apply to the type Rational : == , < , <= , > , >= , + , - , * , and / .

Write a test program to test your class. Hints: Two rational numbers a/b and c/d are equal if a*d equals c*b. If b and d are positive rational numbers, a/b is less than c/d provided a*d is less than c*b. You should include a function to normalize the values stored so that, after normalization, the denominator is positive and the numerator and denominator are as small as possible. For example, after normalization 4/-8 would be represented the same as -1/2.

On the next page is what your program should look like. It should ask for 3 fractions to be inputted, then test each of the operators on the functions. First with inputs 1 and 2, then with inputs 2 and 3. Note that the results displayed are the normalized results

Run1:

```
Enter a fraction in the format integer_numerator/integer_denominator
-25/-9
You entered the equivalent of: 25/9
Enter a fraction in the format integer_numerator/integer_denominator
3/5
You entered the equivalent of: 3/5
Enter a fraction in the format integer_numerator/integer_denominator
-21/9
You entered the equivalent of: -7/3
Testing arithmetic and relational operator overloading
25/9 * 3/5 = 5/3
25/9 + 3/5 = 152/45
25/9 - 3/5 = 98/45
25/9 / 3/5 = 125/27
25/9 < 3/5 = 0
25/9 < 25/9 = 0
25/9 <= 3/5 = 0
25/9 <= 25/9 = 1
25/9 > 3/5 = 1
25/9 > 25/9 = 0
25/9 >= 3/5 = 1
25/9 >= 25/9 = 1
-7/3 * 3/5 = -7/5
-7/3 + 3/5 = -26/15
-7/3 - 3/5 = -44/15
-7/3 / 3/5 = -35/9
-7/3 < 3/5 = 1
-7/3 < -7/3 = 0
-7/3 <= 3/5 = 1
-7/3 <= -7/3 = 1
-7/3 > 3/5 = 0
-7/3 > -7/3 = 0
-7/3 >= 3/5 = 0
-7/3 >= -7/3 = 1
```

Run2:

```
Enter a fraction in the format integer_numerator/integer_denominator
-25/-9
You entered the equivalent of: 25/9
Enter a fraction in the format integer_numerator/integer_denominator
3.5
Bad input format for operator >>. Aborting!
```

You may find it helpful (and it is good practice to) split up your file into a .h file for the class declaration, a .cpp file for the definition of each member function, and a second .cpp file for the test program.

Your grade will be subject to the following condition(s):

- Submission:
  The submission deadline is **5:00PM** on **11/18/16**. You will be penalized in increments of 25% per day late (regardless of the time).

  Submit your code on Canvas. You just need to **upload** your .cpp file. If you organized your code in multiple files then submit a .zip file containing your code.
  Also, PLEASE double check your submission to make sure the file has actually been uploaded.

Your grade will be calculated based on the following (total 10 points)

- Compilation: **2 pts**
  Your code MUST compile in a Linux environment. Since that is the environment in which it will be graded. There is no partial credit available here, either your code compiles or it doesn't.

- Execution/Correctness: **6 pts**
  Your program will be tested based on all the criteria mentioned above. Namely, but not exclusively, overloading the input and output operators, the mathematical and relational operators, and the implementation of all three constructors. This means that if your program seems to work but you have not actually overloaded the operators, you will lose significant points.

- Style/Organization: **2 pts**
  Your code will also be graded on its style. This includes things like using meaningful variable names, useful comments, proper indentation and spacing, and the proper use of functions. Proper use of functions means wrapping up code that is used in multiple parts of your code in a function. All of these things make your code easy to read and maintain. Partial credit will be available here. As a minimum, your code should have a comment at the beginning with your name, date, and a high level but still descriptive overview of what the program does.

  Pay attention to issues of programming style:

  - **use indentation**
  - **comment your code/methods**
  - **use meaningful names for variables**
  - **leave spaces between logical blocks of the code**
  - **use functions properly**