Lab 5:
Dial Zero

Due date: March 17

Assignment

In this lab, you will augment your `Rational` class from Lab 4 with a variety of useful operators and constructors.

1. Copy the `Rational` files from Lab 4 to a new project. **You may need to fix any issues in your original lab when I return it to you.**
2. Modify `Rational` in the following ways. Add:
   (a) A copy constructor that implements an appropriate copy-construction method.
   (b) The assignment operator `operator=`, which is a member function and takes a single `Rational` parameter.
   (c) `operator<<`, which prints the result of the `ToString` method to the lhs stream.
   (d) `operator+`, which takes two `Rational` arguments and returns a `Rational`–object which is the sum of the two arguments. (Hint: use the `Add` method.)
   (e) **Unary** `operator-`, which takes one `Rational` argument and returns a `Rational` that is the negation of the argument.
   (f) **Binary** `operator-`, which takes two `Rational` arguments and returns their difference.
   (g) `operator*`, which takes two `Rational` arguments and returns their product as a `Rational`.
   (h) `operator/`, use your imagination.
   (i) `operator==`, which returns a `bool` indicating if the two `Rational` arguments are equal. (Hint: see `Equals`.)
   (j) `operator!=`, just return the negation of `operator==`.
   (k) Each of `operator<`, `operator>`, `operator<=`, `operator>=`.
3. For each operator above, you should be taking `const Rational&` parameters any time you don't need to modify the parameter. (Which is almost always.)
4. Write a `main` method that demonstrates the behavior of each operator you just wrote. You will need at least two `Rational` objects to do so.

Deliverables

Hand in:

1. A printed copy of your code, **printed from Visual Studio or your IDE when possible.** If you cannot print from your editor, copy your code into Notepad or another program with a fixed-width (monospace) font and print from there. Print **all .h and .cpp files**.
2. The output of your code, as described in step 2 above.
3. **Note**: your code must declare exactly the methods described in this spec, and **nothing more**.