Submit a single source file with your name and assignment number to Canvas

Write a `rational number` class. A rational number is a "ratio-nal" 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 (ADT) 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. Also include a default constructor that initializes an object to 0 (that is, to 0/1).

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. (Both operators can be used on any input/output stream)

Overload all the following operators so that they correctly apply to the type Rational: `==, <, <=, >, >=, +, -, *, and /`. Also overload the unary –.

```
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)
```

Provide a `main` function that thoroughly tests your class implementation. Declare at least two objects that would represent two rational numbers. Let the user input the objects from any input stream. Test the two objects with each of the overloaded functions. Also test the operators with a combination of integers and objects.

The overloaded operations should make using rational numbers a lot easier and more intuitive, have fun testing this class!