Rational Numbers Lab – Day 2

A *rational number* is a number that can be expressed as the quotient of two integers, where the denominator is not zero. For instance, 1/2, 3/4, 40/17, and 5/1 are all rational numbers. C++ does not have a rational data type, and therefore stores all rational numbers as floats or doubles. This can cause problems because some rational numbers, such as 1/3, cannot be represented exactly in decimal notation (at least not with a finite number of digits). In this lab, you will create a simple `rational` class to better represent (positive) rational numbers.

1. Add a method to your class that lets you multiply two rational numbers together. This function should take one rational number argument and **return** the product of the class's rational number with the argument:

   ```
   rational rational::multiply(const rational & other) const
   ```

   Example of how this might be used:

   ```
   rational a(1, 2);
   rational b(3, 4);
   rational c = a.multiply(b); // a and b are unchanged, c is 3/8
   ```

   Hint: This is easier if you let your `reduce()` function do some of the work for you.

   Add code to main to thoroughly test the function after you write it! (Do this for all the functions below!)

2. Add a method to your class that lets you add two rational numbers together. This function should take one rational number argument and **return** the sum of the class's rational number with the argument:

   ```
   rational d(2, 3);
   rational e(3, 4);
   rational f = d.add(e); // d and e are unchanged, f is 17/12
   ```

3. Add a method to your class that lets the user retrieve the rational number as a double. In other words, the user would call this method when they want to represent 1/2 as 0.5 (temporarily, anyway).

   ```
   double rational::as_double() const
   ```

4. You will notice that this code does not do what you would expect:

   ```
   rational a(1, 2), b(1, 2);       // make two rational numbers, both are one half.
   if (a == b)
     cout << "equal!";              // doesn't print equal (doesn't even compile!)
   ```

   Define a method called is_`equal` that tests if two rational numbers are equal:

   ```
   bool rational::is_equal(const rational & other) const
   ```

5. Define a "less than" function:

   ```
   bool rational::is_less_than(const rational & other) const
   ```

6. In your main function, define a vector of rational numbers. Use `push_back` to add a few rational numbers to the vector (just make some up and put them in manually using `push_back`). Write a function to find the largest rational number in the vector and print it out.