Computer Science 250

Project 4 - A Rational Number Class

**Due:** Mon. May 6, 11:30 a.m. (the final exam time). **No late submissions will be accepted.**

A rational number is a number that can be represented as the quotient of two integers. For this project you are to write a C++ class that can be used to represent *non-negative* rational numbers. Although I will only be grading the class that you write, you will probably also want to write a main program to test your class.

Your class should meet the following specifications:

* Rational numbers should be represented by two unsigned ints.
* The class should have three constructors:
	+ A default constructor that initializes the object to 0;
	+ A constructor that takes a single unsigned int parameter N and initializes the object to the rational number N/1;
	+ A constructor that takes two unsigned ints as parameters, representing the numerator and denominator of the rational. This constructor should fail gracefully if the class user attempts to create a rational object with 0 as the denominator.
* It should have input and output member functions with the following signatures and meanings:

/\* Read a string of the form "n/d" from the keyboard, and

 update the Rational object invoking the method to

 represent the indicated number. Assume that both n

 and d are unsigned int values, and that there

 are no spaces between the n, the '/' and the d. \*/

void input();

/\* Write the object to the display in the form n/d.

 The rational should be displayed in simplest form. \*/

void output();

* Overload the following operators as members of the class: +, -, \*, /, ^. Plus, minus, asterisk and slash perform the usual arithmetic operations. You should assume that the user of your class will never attempt a subtraction that results in a rational number less than zero. The ^ operator is a binary operator that raises the rational number (the left operand) to a non-negative integer power (the right operand). If the user requests an undefined operation (for example, division by zero or 0^0), your class should fail gracefully: print a descriptive error message and call exit().
* Your class should overload the insertion and extraction operators (<< and >>) as friends of the class. The insertion operator should print the rational in simplest form. (A rational number is in simplest form if the numerator and denominator have no common factors other than 1.) It is probably easiest to write the insertion and extraction operators so that they use input() and output().

Follow the usual C++ conventions when writing your class, including:

* Place the class definition in the file "Rational.h" and its implementation in "Rational.cc".
* Use reference parameters, const parameters, const functions, and const return values, as appropriate.
* Anything that is not part of the public interface should be in the private section of the class declaration.

**What to turn in:** When you are ready to submit your class, print out a hard copy of the two source files. Then, create a tarball containing these two files. No other files should be included in the tarball. Email me a copy of the tarball as an email attachment. The subject of your email should be CS 250 – Project 4 - <your last name, your first name>.

**Think before you submit. When you send me your electronic submission, make sure it is the version of your class that you wish to have graded.** If the version you submit does not compile (for example) your grade will necessarily receive a substantial penalty.