Programming Assignment

Course #: PROG 2100

**Assignment #2 (“Rational Number Calculator”)**

**Assignment #2**

Define a class for rational numbers. A rational number is a number that can be represented as the quotient of two integers. For example, ½, ¾, 64/2, and so forth are all rational numbers. (By ½ 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 whole Number and define the constructor so that the object will be initialized to the rational number whole Number/1. Include a default constructor that initializes an object to 0 (that is, to 0/1). Overload the output operator <<. Numbers are to be input and output in the form ½, 15/32, 300/400, 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 that loops after each calculation is complete.

**Notes:**

* Your solution must actually use the operators with rational objects when you do the operations. E.g. RationalNumObj = RationalNumObj#1 + RationalNumObject#2.
* The user may not enter decimal numbers as either the numerator or denominator e.g. “1.2/3.6” nor may the solutions be expressed as decimal numbers.
* There is no specific need for a string constructor as this functionality may be done in the overloaded >> operator.
* Inside the code of the overridden >> operator, have it call a validation method before it constructs an object. If the validation method returns invalid, execute a cin.fail and loop back for more input.

**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.
* Since division is normally calculated by flipping one of the rational numbers and then multiplying the two rational numbers together, consider the effect of dividing by 0/#.

**Interface description (i.e. how the user interacts with your program):**

* User inputs entire first fraction (or whole number) as a string.
* User inputs entire second fraction (or whole number) as a string.
* Do **not** ask the user for the operations i.e. +,-,\*,/,==,<,> (this does not apply to bonus)
* Output for the non-bonus solution should list **all the operations** required in the assignment with only the right side of the equation (the answer) displayed normalized.

\*Any Research Component must be implemented (coded), not just “researched” to be graded and it will not affect your “Project functions to specification” mark if it is not included.

NB: Citations - Remember that citations must be provided for any code, algorithm, text or image copied from another source (e.g. the Web, a textbook, an online tutorial, etc.). Not attributing appropriately (plagiarism) or using illegally copied materials (copyright breach) are serious academic offenses. If you are in doubt as to when or how to cite, consult with your instructor and the resources provided by the college.

Source: Modified from **Savitch Absolute C++**

Programming Assignment Rubric

Course #: PROG 2100

**Assignment #2B (“Rational Number Calculator”)**

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student #: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Marking Rubric:**

|  |  |  |
| --- | --- | --- |
| SAAD Marks (10) | | |
| 10% |  | SAAD – produce a testing document that shows all possible inputs and the expected result of processing that input. Include both positive and negative cases. |
| Generic Marks (25) | | |
| 10% |  | Your code should be tested and free of both logic and syntax errors (validation excepted) and includes the SAAD requirements |
| 10% |  | Error checking / validation (i.e. passes tests performed with the intent of causing the program to malfunction) |
| 5% |  | Code is well commented as to functionality, structures, formulas, etc. Formatting and layout of source code is consistent. |
|  | The user interface is functional, appropriate and easy to use. Output is well formatted and easy to follow. |
|  | Variables declared using the appropriate data type, all objects, variables and other program elements are named appropriately. |
| Project Specific Marks (65) | | |
| 5% |  | Rational Class: A class for rational numbers that represents rational numbers as two values (numerator and denominator) of type int. |
| 5% |  | Default Constructor: A default constructor that initializes a Rational object to 0 (that is, to 0/1) |
| 5% |  | Constructor #1: A constructor with two arguments that can be used to set the member variables of a Rational object to any legitimate values |
| 5% |  | Constructor #2: Initializes a Rational object to the rational number ‘whole Number / 1’ |
| 15% |  | Constructor #3: A string constructor that accepts a string e.g. “-1/3” or a whole number e.g. 5 and creates a rational number |
| 5% |  | Normalization: The output of the math operations must be a single normalized fraction i.e. 1 / 2 instead of 2 / 4 |
| 5% |  | Compare Functions: The output of all compare operations (<,>,==) will be a Boolean return. |
| 5% |  | Overloaded mathematical operators i.e. +, - , / , \*, =, += |
| 10% |  | Overloaded Output Operator: e.g. cout << RN |
| 5% |  | A driver that demonstrates the functionality of all constructor and overloaded operators |
| **100** |  | **TOTAL** |