Object Oriented Programming

Home Work 07 **Marks 10**

Instructions

Work on this home work individually. Absolutely NO collaboration is allowed. Any traces of plagiarism would result in a ZERO marks in this homework and possible disciplinary action. Tasks should be coded in **C++**.

Due Date

Paste the solution of the problem (source code .cpp file only) labeled with your complete roll number in SEM – HW 07 and SEA – HW 07 folders for SE Morning and SE Afternoon sections respectively on Tuesday, April 12, 2016 before 05:00 PM. These folders are available at \\printsrv\Teacher Data\Umair Babar\Students.

ADT: RationalNumber

Define a class for **rational numbers**. A rational number is "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 values**.

1. An **integer** named **numerator** displayed above a line or before a slash.
2. An **integer** named **denominator** displayed below or after that line.

Value should only be assigned to **denominator** if it is **non-zero, 1** otherwise.

1. Provide the implementation of **mutators** for **numerator** and **denominator** data members of the class.
2. Provide the implementation of **accessors** for **numerator** and **denominator** data members of the class.

A principle of abstract data type construction is that **constructors** must be present to create objects with legal values. You should provide constructors to make objects out of pairs of **integer** values;

1. A **constructor** that accepts **Rational Number's numerator** and **denominator** as arguments and assigns them to the appropriate member variables.
2. Since every **integer** is also a rational number, 2/1 or 17/1, you should provide a constructor with single **integer** parameter that accept only the value of **numerator** as argument and assign it to the appropriate member variable.

3. Provide the implementation of following member functions and operators
   1. **write** method to write rational numbers in the form 2/3 or 37/51 on the screen.
   2. **read** method to input rational numbers in the form 2/3 or 37/51 from the keyboard.
   3. Overload **plus (+) binary operator** to perform the **addition** of two rational numbers.
   4. Overload **minus (–) binary operator** to perform the **subtraction** of two rational numbers and returns the result.
   5. Overload **multiply (*) binary operator** to perform the **multiplication** of two rational numbers and returns the result.
   6. Overload **divide (/) binary operator** to perform the **division** of two rational numbers and returns the result.
   7. Overload **less than (<) binary operator** to perform the **comparison** of two rational numbers and returns the result.
   8. Overload **equal (==) binary operator** to perform the **comparison** of two rational numbers and returns the result.
   9. Overload **minus (–) unary operator** to convert a rational number into its **negative** form, if it is already not and returns the result.
   10. Overload **logical not (!) unary operator** to return **true** if the rational number is **negative**, **false** otherwise.
4. Once you have written the class, write **main** function and test its functionality by creating some objects of **RationalNumber**.

The formulas will be useful in defining functions:

a/b + c/d	means	(a*d + b*c) / (b*d)
a/b – c/d	means	(a*d – b*c) / (b*d)
(a/b) * (c/d)	means	(a*c) / (b*d)
(a/b) / (c/d)	means	(a*d) / (c*b)
–(a/b)	means	(–a/b)
(a/b) < (c/d)	means	(a*d) < (c*b)
(a/b) == (c/d)	means	(a*d) == (c*d)

Let any sign be carried by the numerator; keep the denominator positive.

NOTE: - No submission will be accepted after the due date and time.

BEST OF LUCK