Object Oriented Programming

Homework 04 **Marks 10**

Instructions

Work on this homework individually. Absolutely NO collaboration is allowed. Any traces of plagiarism would result in ZERO marks in this homework and possible disciplinary action. Task should be coded in **C++**. You are strictly ***NOT ALLOWED*** to include any additional data-members/functions/constructors in your class. Write the ***main*** function first and keep testing the functionality of each function once created.

Due Date

Paste the solution *(source code **.cpp** file only)* labeled with your complete roll number e.g., BSEF21M000 in OOP BSEMF21 HW 04 folder till 05:00PM Monday, October 03, 2022. The folder is available at \\printsrv\Teacher Data\Umair Babar\Students.

ADT: RationalNumber

Write 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** value.

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. Overload the following operators.
   1. **Stream-insertion operator (<<)** to write rational numbers in the form 2/3 or 37/51 on the screen.
   2. **Stream-extraction operator (>>)** to input rational numbers in the form 2/3 or 37/51 from the keyboard.
   3. **Plus (+) binary operator** to perform the **addition** of two rational numbers.
   4. **Minus (–) binary operator** to perform the **subtraction** of two rational numbers and returns the result.
   5. **Multiply (*) binary operator** to perform the **multiplication** of two rational numbers and returns the result.
   6. **Divide (/) binary operator** to perform the **division** of two rational numbers and returns the result.
   7. **Less-than (<) binary operator** to perform the **comparison** of two rational numbers and returns the result.
   8. **Equal (==) binary operator** to perform the **comparison** of two rational numbers and returns the result.
   9. **Minus (–) unary operator** to convert a rational number into its **negative** form, if it is already not and returns the result.
   10. **Logical not (!) unary operator** to return **true** if the rational number is **negative**, **false** otherwise.

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**.

Failure to a abide by the submission instructions will cause a penalty of two marks.
No submission will be accepted after the due date and time.

BEST OF LUCK