COMSC 200

Summer 2024

Programming Assignment 9

Worth 12.5 points (1.25% of your grade)

DUE: Monday, 7/8/24 by 11:59 P.M. on Canvas

Late Pass Deadline: Thursday, 7/11/24 by 11:59 P.M. on Canvas

You need to start by downloading the following FIVE (5) files from Canvas: app_prog1.cpp, app_prog2.cpp, app_prog3.cpp, app_prog4.cpp, and app_prog5.cpp

Your solution that you submit should consist of three (3) files:

RationalNumber.h (class specification file)

RationalNumber.cpp (class implementation file)

And the first-name_last-name.pdf file should contain your five (5) sample runs.

The application programs have already been completed for you. **Your RationalNumber class should work with the application programs that has been given to you.**

Please continue to use the same **naming convention** as before, where each filename should contain both your first name and your last name. If your first name is “James” and your last name is “Smith”, then your header file should be named James_Smith_ RationalNumber.h, and your cpp file should be named James_Smith_ RationalNumber.cpp

Comments – worth 1.25 points (10%) of your programming assignment grade:

Your program should have at least **ten (10)** different detailed comments explaining the different parts of your program. Each individual comment should be, at a minimum, a sentence explaining a particular part of your code. You should make each comment as detailed as necessary to fully explain your code. You should also number each of your comments (i.e., comment 1, comment 2, etc.).

Sample Runs – worth 1.25 points (10%) of your programming assignment grade:

You should submit screenshots of at least **five (5)** different sample runs of your program. You should also number each of your sample runs (i.e., sample run 1, sample run 2, etc.). **NOTE: Your sample runs should be different from my sample runs shown in this write-up for the programming assignment.** To generate each sample run, you can modify the app_prog1.cpp file that I gave you and change the values for the c and d RadtionalNumber objects for each sample run. Each sample run should consist of a screenshot of the RationalNumber object from the application program and the corresponding output on the console screen:

```
RationalNumber c( 7, 3 ), d( 3, 9 ), x;
```

```
7/3 + 1/3 = 8/3
7/3 - 1/3 = 2
7/3 * 1/3 = 7/9
7/3 / 1/3 = 7
7/3 is:
  > 1/3 according to the overloaded > operator
  >= 1/3 according to the overloaded < operator
  >= 1/3 according to the overloaded >= operator
  > 1/3 according to the overloaded <= operator
  != 1/3 according to the overloaded == operator
  != 1/3 according to the overloaded != operator
```

For this programming assignment you will be implemented a RationalNumber class. In mathematics, a **rational number** is a number that can be represented as a fraction, where both the numerator and the denominator in the fraction are integers (i.e., whole numbers). Every integer is a rational number, because the denominator in the fraction can be 1 (i.e., 7 can be represented as 7/1). Some (but not all) floating pointer numbers (i.e., real numbers) are rational numbers as well. The floating point number 2.5 is also a rational number, because it can be represented as the fraction 5/2.

Create a class RationalNumber(fractions) with these capabilities:

a. Implement a constructor that receives two integer parameters (the first is the numerator, and the second is the denominator) that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators.
b. Implement a printRational member function that prints out a RationalNumber object.
c. Overload the addition (+), subtraction (-), multiplication (*), division (/) operators, and assignment (=) operators for this class.
d. Overload the relational and equality (<, >, <=, >=, ==, and !=) operators for this class.

Sample run 1 (using the **app_prog1** application program):

```
RationalNumber c( 7, 3 ), d( 3, 9 ), x;
```

```
7/3 + 1/3 = 8/3
7/3 - 1/3 = 2
7/3 * 1/3 = 7/9
7/3 / 1/3 = 7
7/3 is:
  > 1/3 according to the overloaded > operator
  >= 1/3 according to the overloaded < operator
  >= 1/3 according to the overloaded >= operator
  > 1/3 according to the overloaded <= operator
  != 1/3 according to the overloaded == operator
  != 1/3 according to the overloaded != operator
```

Sample run 2 (using the **app_prog2** application program):

```
RationalNumber c( 4, 3), d( 5, 2), x;
```

```
4/3 + 5/2 = 23/6
4/3 - 5/2 = -7/6
4/3 * 5/2 = 10/3
4/3 / 5/2 = 8/15
4/3 is:
  <= 5/2 according to the overloaded > operator
  < 5/2 according to the overloaded < operator
  < 5/2 according to the overloaded >= operator
  <= 5/2 according to the overloaded <= operator
  != 5/2 according to the overloaded == operator
  != 5/2 according to the overloaded != operator
```

Sample run 3 (using the **app_prog3** application program):

```
RationalNumber c(1, 2), d(6, 12), x;
```

```
1/2 + 1/2 = 1
1/2 - 1/2 = 0
1/2 * 1/2 = 1/4
1/2 / 1/2 = 1
1/2 is:
  <= 1/2 according to the overloaded > operator
  >= 1/2 according to the overloaded < operator
  >= 1/2 according to the overloaded >= operator
  <= 1/2 according to the overloaded <= operator
  == 1/2 according to the overloaded == operator
  == 1/2 according to the overloaded != operator
```

app_prog4 application program):

Sample run 4 (using the

```
RationalNumber c(2, 7), d(5, 11), x;
```

```
2/7 + 5/11 = 57/77
2/7 - 5/11 = -13/77
2/7 * 5/11 = 10/77
2/7 / 5/11 = 22/35
2/7 is:
  <= 5/11 according to the overloaded > operator
  < 5/11 according to the overloaded < operator
  < 5/11 according to the overloaded >= operator
  <= 5/11 according to the overloaded <= operator
  != 5/11 according to the overloaded == operator
  != 5/11 according to the overloaded != operator
```

Sample run 5 (using the **app_prog5** application program):

app_prog5 application program):

```
RationalNumber c(9, 3), d(14, 2), x;
```

```
3 + 7 = 10
3 - 7 = -4
3 * 7 = 21
3 / 7 = 3/7
3 is:
  <= 7 according to the overloaded > operator
  < 7 according to the overloaded < operator
  < 7 according to the overloaded >= operator
  <= 7 according to the overloaded <= operator
  != 7 according to the overloaded == operator
  != 7 according to the overloaded != operator
```