CISC 3142
Programming Paradigms in C++
Lab #7
Operator Overloading

How to Develop and Submit your Labs

Make sure you read the sections in the above on how to interpret CodeLab string comparison feedback

Lab 7 — Rational with Operator Overloading

Overview

Like it says... the `Rational` class in C++, but this time using the standard arithmetic and equality operators.. As before, I will be supplying a test program (`rational_test.cpp`) as well as an exception class (`rational_exception.h`).

Program Organization

Same file organization as before. You will be submitting `rational.h` and `rational.cpp` as two separate exercises in CodeLab. As I will be supplying the other, the function signatures in the `.h` and `.cpp` files must match (which is the whole reason I;m doing it this way — to force you to use the correct signatures). **All** the function definitions (bodies) should go into the `.cpp`, i.e., none should be coded inline in the class declaration in the `.h`.

Member and Helper Functions

I am following the usual rules:

- 'in-place' and unary operators are members
- other binary operators are non-member helper functions

Notes

- Much of this can be accomplished by changing the function names to operators
- There are some significant changes in the class structure … in particular the removal of the binary operators from the class and making them non-member helpers.
- As before, you can (and now should) leverage the functionality of the arithmetic operator and their corresponding *inPlace* functions
    - In this exercise yo should define the 'in-place' (e.g. += operator first, since it's a member and has access to the class representation, and then leverage that functionality to implement the corresponding non-member binary operator (in this case +).
- You are now coding the << operator. You can either continue to use your `print` function or move its

functionality into the operator.

Submitting to Codelab

As in Labs 4.1 and 4.2, this lab is broken up in CodeLab into multiple 'subexercises':

- 7.1 - Submit your `rational.h`
- 7.2 - Submit your `rational.cpp`

A Word About the Compiler Feedback in CodeLab

In all likelihood, you will make a mistake on at least one function signature (by mistake I mean not matching my corresponding signature in the corresponding `h` / `.cpp` file; it may also be the case that *I* made a mistake in the signature). In the event of such (interface) errors, the compiler will complain about the two functions signatures not matching. In order for you to figure out your error, you should know which signature is your and which is mine, and where the compiler will be doing the complaining. In 4.1.1, the compiler will encounter the `.h` file first (because the `#include` is encountered at the top of the `.cpp` file), and thus the error will be manifested when it reaches *my* signature (in the `.cpp` which won't match. Conversely, in 4.1.2 — where *I* am providing the `.h` file, the error will again appear in the `.cpp` signature, which is yours this time.

This exercise addresses many of the topics covered in the 'Preliminaries' lectures, this time in C++

- Constructor overloading
- Scoping, in particular class scope
- Leveraging functionality, and maintaining semantic consistency
- Mutable vs immutable semantics and methods
- Asymmetric operands (receiver vs argument) in methods corresponding to binary operations
- One and two argument operations

Code Used in this Lab