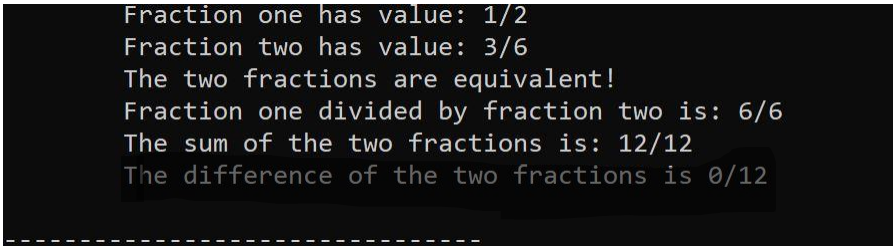
Lab 12: The Fraction Calculator

Objectives:

Practice operator overloading in C++ with classes.

Background:

In this lab, you are going to work with the *fraction* class and implement some of the essential operators using overloading method. The output that you are trying to achieve looks like this:



First, you need to declare two fraction objects and initialize their values to 1/2 and 3/6 accordingly.

Then, you need to output their corresponding values using the insertion operator.

Next, you need to check whether the two fractions are equal using the == operator overload.

Next, you need to perform division of the two fractions using the / operator overload. For the division, you want to divide “fraction one” by “fraction two”.

Next, you need to perform addition on the two fractions using the + operator overloads.

Specifications:

For this lab, you do not need to use multiple files. The class definition can go before the main function, and the function implementations can be placed after the main function.

The fraction class:

Here is a snippet of how the class definition for the fraction class can look like:

```
class fraction
{
  public:
    // Constructors (if applicable)
    // Getters and Setters (if applicable)

    // Public interface functions
    fraction operator /(const fraction & rhs);

    // Friend functions
    friend ostream & operator <<(ostream & os, const fraction & rhs);
    friend bool operator ==(const fraction & lhs, const fraction & rhs);
    friend fraction operator +(const fraction & lhs, const fraction & rhs);
    friend fraction operator -(const fraction & lhs, const fraction & rhs);
  private:
    int m_num;
    int m_den;
};
```

For the constructors, public setters and getters, you can decide if you need to implement any of them. You may not have to, but you have the freedom to.

1. For the insertion operator overload, you need to make it a **non-member function**, thus the *friend* keyword which signifies a friend function.
2. For the == operator overload, you need to make it a **non-member function**, thus the *friend* keyword which signifies a friend function. A hint on how to implement this function overload is this:

   Given two fractions a/b and c/d, what would this relationship tell you?

   ad == bc?
3. For the division operator overload, you need to make it a **member function**, which means that there will be a calling object.

   If we are to follow the programming guideline, this division operator overload should also be made as a non-member function, but we implement it as a member

function simply to demonstrate the point that you can make this as a member function overload and thus would be a good coding exercise.

Here is a hint for implementing / overload:

If you want to perform a/b divided by c/d, then it is identical to performing the following:

a/b * d/c

Requirements

1. Proper file header
2. Proper formatting (e.g. indentation)
3. Have the correct program output and meet all of the specifications
4. Your answers should be equivalent to mine.