**Lab 4: Creating a Complete C++ Class**

# Overview

In this lab you will ceate a complete C++ class, which will include:

1. Class variables
2. Overloaded constructors
3. Additional class methods

In addition, you will write a program that uses the new class.

# Class Description

In this lab you will create a complete class to handle fractions. The fraction class will have data members to hold the numerator and denominator, both of which will be integers.

Your class must have the following member functions:

1. Print a fraction to the screen using a function called Output().
2. Compute the reciprocal of a fraction using a function called Reciprocal().

Additionally your class must have two constructors:

1. A default constructor that initializes the fraction to 1/1.
2. A second constructor that allows the user to initialize the numerator and denominator using the syntax Fraction frac(numerator, denominator). This constructor must also check that the denominator is not zero.

You will create a source file lab4.cpp to contain your main() function, but the class declaration and definitions must be in the files Fraction.h and Fraction.cpp, respectively.

Here is skeleton code for lab4.cpp:

#include <iostream>

#include "Fraction.h"

using namespace std;

int main()

{

// Create 3 fractions:

// - Read the first fraction from the keyboard using cin.

// Find the reciprocal of this fraction and print it to the screen.

// - The second fraction will be created with default values. You

// should also print this to the screen.

// - Attempt to create a fraction with a denominator of zero, which

// should print an error

return 0;

}

Starting The Class

First create your skeletal class structure and declare its data members. You should at least have members to hold the numerator and denominator. Be sure to use appropriate meaningful names and to declare data members as private, following class conventions.

Next declare the functions that will access the class members. These should be your output and reciprocal functions as well as your constructors.

Reciprocal and Output Functions

Now you will write the reciprocal and output functions. The reciprocal function should return a new fraction object with the numerator and denominator swapped. For example:

Fraction frac(1, 2); // frac is 1/2

Fraction newFrac = frac.Reciprocal(); // newFrac is 2/1

Your output function must print the numerator and denominator of the fraction separated by a forward slash. For example:

Fraction frac(2, 3);

frac.Output(); // produces the output "2/3"

Consider whether the functions should be const.

Constructors

Now create your two class constructors. The first constructor must intialize the fracton to 1/1. The second allows the fraction to be intialized to any value. For example, the fraction 12/17 could be created by Fraction frac(12, 17). Remember that the second constructor must check whether the denominator is zero and, if so, print an error message.

Building your Program

You can build your program with the following command:

g++ -ansi -Wall lab4.cpp Fraction.cpp -o lab4

There are really two distinct steps in the program build:

1. lab4.cpp and Fraction.cpp are *compiled* to produce the object files lab4.o andFraction.o.
2. The objects files are *linked* to produce the executable lab4.

We can execute these separate steps explicitly as follows:

g++ -c lab4.cpp

g++ -c Fraction.cpp

g++ lab4.o Fraction.o -o lab4

Understanding this explict form of the build process is necessary to be able to write makefiles. Complete the following makefile, which uses the explicit build steps.

CPPFLAGS = -ansi -Wall

lab4: lab4.o Fraction.o

g++ lab4.o Fraction.o -o lab4

lab4.o: lab4.cpp Fraction.h

WHAT GOES HERE?

Fraction.o: WHAT GOES HERE...

AND HERE?