ECS40 Winter 2017 2017-01-19

Homework 1 due Tue 2017-01-31 at 20:00

Use the handin directory hw1 to submit your work

Fraction calculator

Description

Create a simple fraction calculator that can add and subtract any number of fractions and writes the answer as a reduced fraction.

Your program will read input from **`stdin`** and write output on **`stdout`**.
A fraction is represented as the sequence:

```
a/b
```

where **`a`** and **`b`** are integers and any amount of white space characters `' '` (including none) can separate **`a`** from `'/'` and `'/'` from **`b.`**

Input consists of an expression made of fractions separated by the operators `'+'` or `'-'.`
The number of fractions in the expression is arbitrary. Each of the following 6 lines is an example of valid input expression:

```
1/2 + 3/4
1/2-5/7+3/5
355/113
3       /9- 21 /   -7
4/7-5/-8
-2/-3 +7 /5
```

Note that the numerator and/or denominator of a fraction given as input may be negative.

The input will consist of a single expression on a single line terminated by a **`\n`** character.

The output should consist of a single, irreducible fraction written as

```
c/d
```

where **`c`** is a signed integer and **`d`** is a positive integer (i.e. the denominator cannot be negative in the output). The numbers **`c`** and **`d`** should not have any common factors (apart from 1).

Error processing and special cases

You can expect the input to consist of complete expressions, i.e. there will be no incomplete or missing fractions in the input. Fractions with a zero denominator may appear in input, and must cause the following error message to be printed on **`stdout`**:

`Error: denominator is zero`

with *no other output*.

If the answer is an integer, the program should only print that integer and not a fraction

Example: input= **`3/2 + 4/8`** , output = **`2`**
Examples of input and corresponding output will be provided on the web site site web.cs.ucdavis.edu/~fgygi/ecs40 . Your output should exactly match the output in the example files.

Implementation

The main program, called **`calculator.cpp`** , is provided. Your project must include a class **`Fraction`** that encapsulates all functions related to the processing of fractions. The **`>>`** and **`<<`** operators must be overloaded and used to read and write fractions. The presence of a zero denominator in input should be handled by throwing an exception of type **`invalid_argument`** defined in **`<stdexcept>`**. The class **`Fraction`** must be defined in a source file **`Fraction.cpp`** and declared in a header file **`Fraction.h`**. The implementation of the member functions should be defined in the file **`Fraction.cpp`**. The class Fraction must include a constructor **`Fraction::Fraction(int a, int b)`**. The internal representation of the fraction should be in reduced form, i.e. using a pair of numbers that have no common factors. Constructors and other member functions must ensure that the fraction is always reduced. Use Euclid's algorithm to simplify fractions to reduced form. An example of a C implementation of this algorithm is provided on the web site web.cs.ucdavis.edu/~fgygi/ecs40 in the example programs of Lecture 1. The operators **`'+'`**, **`'-'`** and **`'='`** must be overloaded and defined. Member functions **`getNum()`** and **`getDen()`** should be implemented, returning the (reduced) numerator and denominator of the fraction. It should be possible to use the class **`Fraction`** in the program **`useFraction.cpp`** which **`#include`**s the header **`Fraction.h`**. The program **`useFraction.cpp`** is provided on the web site web.cs.ucdavis.edu/~fgygi/ecs40. You should not modify that program.

Submission

Create a tar file named **`hw1.tar`** containing the files **`calculator.cpp Fraction.cpp Fraction.h useFraction.cpp`** and **`Makefile`**. Do not compress the tar file. The files **`calculator.cpp`** and **`useFraction.cpp`** must be identical to the files provided. The Makefile must contain a target **`calculator`** , a target **`useFraction`** and a target **`all.`** The target **`all`** should appear first and will be used to build both executables. The Makefile should include the necessary definition to compile C++ files with the **`-Wall`** option. Include your name and Student ID in a comment at the top of each file (except **`calculator.cpp`** and **`useFraction.cpp`**). Submit your project using:

`$ handin cs40 hw1 hw1.tar`