02393 C++ Programming Exercises

Assignment 6

Hand-in via `https://dtu.codejudge.net/02393-e17/assignment/`

A class for fractions of integers The goal of the exercise of this week is to implement a class of fractions of integers supporting some basic operations like addition and multiplication. You can use the following interface for the class as a starting point and modify it at will:

```
class fraction {

private:
    // Internal representation of a fraction as two integers
    int numerator;
    int denominator;

public:
    // Class constructor
    fraction(int n, int d);

    // Methods to update the fraction
    void add(fraction f);
    void mult(fraction f);
    void div(fraction f);

    // Display method
    void display(void);

};
```

The main idea above is that a fraction $\frac{a}{b}$ is represented by two integers (the numerator a and the denominator b) and several methods are provided to support arithmetic operations on fractions. For example:

`fraction(int n, int m)` constructs the fraction $\frac{n}{m}$;

`void add(fraction f)` updates a fraction by adding fraction f to it.

`void mult(fraction f)` updates a fraction by multiplying it by fraction f.

`void div(fraction f)` updates a fraction by dividing it by fraction f.

Your program should read sequences of simple expressions from `cin` of the form: `a / b + c / d`, `a / b * c / d` or `a / b div c / d` and provide the result as a simplified fraction. For example, if the input is

```
1 / 4 + 1 / 2
```

your program should provide the following output to `cout`

```
3 / 4
```

which results from $\frac{1\cdot 2+1\cdot 4}{4\cdot 2} = \frac{6}{8}$ which after simplification is $\frac{3}{4}$.

Challenge The suggested internal representation poses some limits on the actual domain of integral numbers that the fraction class can cover. For example, the largest number would be $\frac{\texttt{INT\_MAX}}{1}$ and the smallest positive fraction would be $\frac{1}{\texttt{INT\_MAX}}$. A possible remedy to mitigate this would be to exploit the fact that every integer can be represented by a product of prime numbers (see e.g. `https://en.wikipedia.org/wiki/Fundamental_theorem_of_arithmetic`). This was indeed what we exploited in assignment 2. For example

$$137200 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 5 \cdot 5 \cdot 7 \cdot 7 \cdot 7$$

To avoid repeating the same prime number many times, we could equivalently represent 137200 with exponentiation of prime numbers:

$$137200 = 2^4 \cdot 5^2 \cdot 7^3$$

We could thus represent all the exponents in an array of exponents, where the i-th element of the array represents how many times the i-th prime number occurs.[1] So the representation of 137200 would be

array index	0	1	2	3	4	5	...
array	4	0	2	3	0	0	...
corresponding prime number	2	3	5	7	11	13	...

Arithmetic operations can also exploit such representation. For instance, the multiplication of two numbers in such representation can be obtained by adding the corresponding exponents. For instance, multiplying the factorizations of 12 and 10 would be done as follows

primes	2	3	5
12	2	1	0
10	1	0	1

Adding exponents gives:

120	3	1	1

Division can be obtained in a similar manner, by substracting the exponents. Addition may be more involved.

The challenge is to implement the class of fractions with this representation technique.

[1] Formally, the factorization of a natural number $n > 0$ consists of the (uniquely determined) exponents $k_1, k_2, \ldots$ such that $n = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots$ where $p_1, p_2, \ldots$ are the prime numbers.