Computing 2 Homework Problems

Problem Set 3 (More Fractions)

Create a C++ class `Fraction`, comprising two private integer members called `num_` (for numerator) and `den_` (for denominator). The class's default constructor should set both members to a default value of 1.

There should also be a parameterized constructor that constructs an object of type `Fraction` given values for the numerator and denominator. The constructor must also check on the denominator's value to make sure that it is not zero. If it is, an error is to be reported followed by `exit(1)`.

The stream insertion operator (<<) is to be overloaded such that objects of type `Fraction` can be output in the format numerator/denominator.
For example:

```
f = Fraction(3, 4);
cout << "f = " << f;
```

This should yield:

f =3/4

Also overload the operators +,-,*,/ as for addition, subtraction, multiplication and division of two `Fraction` objects as well as with numbers of fundamental data type `int`. Think about how you can use a conversion constructor to make your life easier.

This is how you could carry out the computations:

- Sum of two fractions: $\frac{a}{b}+\frac{c}{d}=\frac{ad+cb}{bd}$
- Difference of two fractions: $\frac{a}{b}-\frac{c}{d}=\frac{ad-cb}{bd}$
- Product of two fractions: $\frac{a}{b}*\frac{c}{d}=ac/bd$
- Division of two fractions: : $\frac{a}{b}/\frac{c}{d}=ad/bc$
- Product of a fraction and a Integer number : $\frac{a}{b}*c=\frac{ac}{b}$
- Division of a fractions by a Integer number : $\frac{a}{b}/c=\frac{a}{bc}$

When carrying out the computations above, you may end up with fractions that can be reduced. Reduction can be done by using the greatest common divisor (gcd). Here is a utility function for that.

```
void Fraction::normalize()
{
   if ( num_ == 0) { den_ = 1; return ; }

   int sign = 1;
   if ( num_ < 0) { sign = -1; num_ = -num_; }
   if ( den_ < 0) { sign = -sign ; den_ = -den_; }

   long gcd   = static_cast<long>(num_); // greatest common divisor
   long value = static_cast<long>(den_);
   while ( value != gcd ) // stop when the GCD is found
   {
      if ( gcd > value )
         gcd = gcd - value ; // subtract smaller number from the greater
      else
           value = value - gcd ;
   }

   num_ = sign * ( num_ / gcd ) ;
   den_ = den_/ gcd ;  // denominator is always positive
}
```

Add this function as a helper function of the class.

Write a driver program that produces the following output:

```
7/3 + 1/3 = 8/3
7/3 - 1/3 = 2
7/3 * 1/3 = 7/9
3 + 1/3 = 10/3
3 - 1/3 = 8/3
3 * 1/3 = 1
3 / 1/3 = 9
```