```
//Luis Ortiz Baca
//Danny Vu
/*************************************************************

 * The implementation file fraction.cpp defining the instance *

 * member functions and helper functions for  Fraction class  *

 *************************************************************/



#include <iostream>
#include <cmath>
#include <cassert>
#include "fraction.h"

using namespace std;



/*************************************************************

 * The parameter constructor gets values for the numerator    *

 * and denominator, initializes the object, and normalizes the*

 * value of the numerator and the denominator according to the*

 * conditions defined in the class invariant. If denominator is 0, then set it to 1.            *

 *************************************************************/

Fraction::Fraction(int num, int den) {
   numer = num;
   if (den == 0) {
      denom = 1;
   }
   else {
      denom = den;
   }
}
```

```
 /*************************************************************
  * The default constructor creates a fraction as 0/1. It does *
  * not need validation.                                       *
  *************************************************************/
Fraction::Fraction() {
    numer = 0;
    denom = 1;
}


  /*************************************************************
   *The copy constructor creates a new fraction from an exisiting*
   * object. It does not need normalization because the source  *
   * object is already normalized.                              *
   *************************************************************/
Fraction::Fraction(const Fraction& f1) {
    numer = f1.getNumer();
    denom = f1.getDenom();
}


   /*************************************************************
    * The destructor simply cleans up a fraction for recycling.  *
    *************************************************************/


Fraction :: ~Fraction()

{
```

```
}

/*************************************************************
 * The getNumer function is an accessor function returning the*
 * numerator of the host object. It needs the const modifier  *
 *************************************************************/

int Fraction::getNumer() const
{
    return numer;
}

/*************************************************************
 *The getDenum function is an accessor function returning the *
 *denominator of the host object. It needs the const modifier.*
 *************************************************************/

int Fraction::getDenom() const
{
    return denom;
}
```

```
/************************************************************
 * The print function is an  accessor function with a side    *
 * effect that displays the fraction object in the form x/y.  *
 ************************************************************/

void Fraction::print() const
{
    cout << numer << "/" << denom << endl;
}

/************************************************************
 *The setNumer is a mutator function the changes the numerator*
 *of an existing object. The object needs normalization.      *
 ************************************************************/

void Fraction::setNumer(int num)
{
    numer = num;
    normalize();
}
```

```
/*************************************************************
 * The setDenom is a mutator function that changes denominator*
 * of an existing object. The object needs normalization.     *
 *************************************************************/

void Fraction::setDenom(int den)
{
    denom = den;
    normalize();
}

/*************************************************************
 * Normalize function takes care of three fraction invariants *
 *************************************************************/

void Fraction::normalize()
{
    // Handling a denominator of zero
    if (denom == 0)
    {
        cout << "Invalid denomination. Need to quit." << endl;
```

```
        assert(false);
    }

    // Changing the sign of denominator
    if (denom < 0)
    {
        denom = -denom;
        numer = -numer;
    }

    // Dividing numerator and denominator by gcd
    int divisor = gcd(abs(numer), abs(denom));
    numer = numer / divisor;
    denom = denom / divisor;
}

/*************************************************************
 * The gcd function finds the greatest common divisor between *
 * the numerator and the denominator.                         *
 *************************************************************/

int Fraction::gcd(int n, int m)
```

```
{
    int gcd = 1;

    for (int k = 1; k <= n && k <= m; k++)
    {
        if (n % k == 0 && m % k == 0)
        {
            gcd = k;
        }
    }
    return gcd;

}
```