Driver

```cpp
//Joon Im
//Marie Payad
//Demo:8:52
#include "Fraction.h"
#include <iostream>
using namespace std;

int main()
{
  // Instantiation of some objects
  Fraction fract1 ;
  Fraction fract2 (14, 21);
  Fraction fract3 (11, -8);
  Fraction fract4 (fract3);
  Fraction fract5(2,0);

  // Printing the object
  cout << "Printing four fractions after constructed: " << endl;
  cout << "fract1: ";
  fract1. print();
  cout << "fract2: ";
  fract2. print();
  cout << "fract3: ";
  fract3. print();
  cout << "fract4: ";
  fract4. print();
  cout << "fract5: ";
  fract5.print();

  // Using mutators
  cout << "Changing the first two fractions and printing them:";
  cout << endl;
  fract1.setNumer(4);
  cout << "fract1: ";
  fract1.print();
  fract2.setDenom(-5);
  cout << "fract2: ";
  fract2.print();

  // Using accessors
  cout << "Testing the changes in two fractions:" << endl;
  cout << "fract1 numerator: " << fract1.getNumer() << endl;
  cout << "fract2 numerator: " << fract2.getDenom() << endl;
  return 0;
}
```

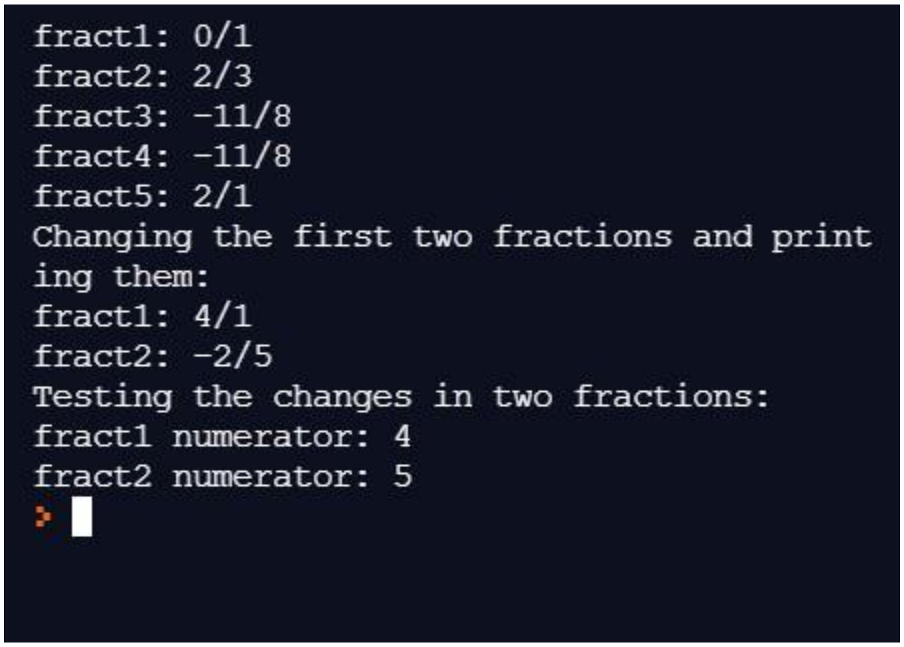
Fraction.h

```cpp
#include <iostream>

using namespace std;

class Fraction
{
  // Data members
  private:
    int numer;
    int denom;
  // Public member functions
  public:                                                                                         // Constructors
    Fraction (int num, int den);
    Fraction ();
    Fraction (const Fraction& fract);
    ~Fraction ();
  // Accessors
  int getNumer () const;
  int getDenom () const;
  void print () const;
  // Mutators
  void setNumer (int num);
  void setDenom (int den);
  // Helping private member functions
  private:
    void normalize ();
    int gcd (int n, int m);
};
```

Fraction CPP

```
#include <iostream>
#include <cmath>
#include <cassert>
#include "Fraction.h"

using namespace std;

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

Fraction::Fraction()
: numer (0), denom (1)
{
}

Fraction::Fraction (const Fraction& fract)
: numer (fract.numer), denom (fract.denom)
{
}

Fraction::~Fraction ()
{
}

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

```cpp
}

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

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

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

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

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;
}

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;
}
```



```
fract1: 0/1
fract2: 2/3
fract3: -11/8
fract4: -11/8
fract5: 2/1
Changing the first two fractions and print
ing them:
fract1: 4/1
fract2: -2/5
Testing the changes in two fractions:
fract1 numerator: 4
fract2 numerator: 5
>
```