Lab 4

Turn In:

1. Coding Assignment – Due Thursday, February ???, 2013
    a) For each exercise, a hardcopy package must be generated to include the following items:
        - Cover Sheet (see the sample copy include in lecture note)
        - Exercise/problem statement
        - Copy of program (named as **`cis27Spring2013YourNameLab4Ex1)`**
        - Copy of output (copy and paste from output screen as possible)

    b) Submitting in class one hard copy package for each exercise; and
    c) Emailing your work as follows,
        - One message for each exercise.
        - Attaching the source file (program) that was created in part (a).
        - The SUBJECT line of the message should have one of the following lines:
          `CIS 27 Spring 2013 Your Name : Lab 4 – Exercise #1`
          Or,
          `cis27Spring2013YourNameLab4Ex1`

2. Q.E.D.

****

1. Coding Assignment

Exercise #1

Write a menu program to have the display below,

```
CIS 27 – C Programming
Laney College
Your Name

Assignment Information --
  Assignment Number:  Lab 04,
                      Coding Assignment -- Exercise #1
  Written by:         Your Name
  Submitted Date:     Due Date
```

While working with fractions, one would like to display the results as ratios of integers. The computations need to be performed with fractions to produce the results that are fractions in reduced form.

Write a program named `cis27Spring2013YourNameLab4Ex1.c` that will allow simple binary arithmetic operations of addition, subtraction, multiplication, and division working with several pairs of common fractions.

A `main()` should call appropriate function(s) to perform the required operation(s) requested by the user – A **Menu Program** should be implemented to test the desired functions .

> Note that fraction data/value **must** be declared as a **`struct Fraction`** of two elements, which are integers of `num` and `denom`. These Fraction objects must have their negativity to be taken to the denominators.

The functions should handle all details and logic required by the operations. The functions are named as follows,

`addFractionYourName()`

`subtractFractionYourName()`

`multiplyFractionYourName()`

`divideFractionYourName()`

The above functions must satisfy the following requirements:

a) All arguments must be passed to each function appropriately; and

b) The return value from each function should be a pointer to a dynamic **`struct Fraction`**; and

c) Each function must include appropriate logic to avoid cases (exceptions) such as divided by zero (0), incorrect values of numerator/denominator, etc.; and

d) Other possible preventive logic.

To test your program, use the follow pairs of common fractions for each of the four binary operations:

```
( 3 / 4, 5 / 12 ),
( -3 / 7, 4 / 9 ),
( 2 / 11, 5 / 101 ),
( 6 / 17, - 8 / 15 )
```