Lab 6

Turn In:

1. Coding Assignment – Due Txxxsday, xxx xx, 2014
    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 **`cis27Spring2014YourNameLab6Ex1)`**
        - 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 2014 Your Name : Lab 6 – Exercise #1`
          Or,
          `cis27Spring2014YourNameLab6Ex1`

2. Q.E.D.

****

1. Coding Assignment

Exercise #1

1. Write a menu program to have the display below,

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

Assignment Information --
  Assignment Number:  Lab 06,
                      Coding Assignment -- Exercise #1
  Written by:         Your Name
  Submitted Date:     Due Date
```

2. You are going to work with polynomials that have fraction as coefficients and integers as powers/exponents. The polynomials will be created as linked lists with the information and specifications given below (you should attach YourName at the ends of the structures and functions below).

```
struct Fraction {
  int num;
  int denom;
};

struct PTerm {
  int ex;
  struct Fraction coe;
};

typedef struct PTerm Term;
typedef struct PTerm* TermAddr;

struct PolyListNode {
  TermAddr termAddr;
  struct PolyListNode* next;
};

typedef struct PolyListNode PolyNode;
typedef struct PolyListNode* PolyNodeAddr;
typedef struct PolyListNode* PolyList;
typedef PolyList* PolyListAddr;

TermAddr createTerm(void); // quiz on Tuesday
TermAddr createTerm01(struct Fraction coe, int ex);

PolyNodeAddr createPolyNode(void);
PolyNodeAddr createPolyNode01(TermAddr);

int insertPolyNode(PolyList*, PolyNodeAddr);

int removePolyNode(PolyList*, int order);
```

And

```
struct ExpressionTerm {
  TermAddr termPtr;
  char op;
};

typedef struct ExpressionTerm ExprTerm;
typedef ExprTerm* ExprTermAddr;
```

```
typedef ExprTerm* ExprTermPtr;

struct ExpressionListNode {
  ExprTermAddr termAddr;
  struct ExpressionListNode* next;
};

typedef struct ExpressionListNode ExprNode;
typedef struct ExpressionListNode* ExprNodeAddr;
typedef struct ExpressionListNode* ExprList;
typedef ExprList* ExprListAddr;

struct ExpressionNodeStack {
  int size;
  ExprNodeAddr top;
  ExprList* exprListAddr;
};

typedef struct ExpressionNodeStack ExprNodeStack;

ExprNodeAddr popEN(ExprNodeStack*); // MUST BE IMPLEMENTED BY
                                    // STUDENT
int pushEN(ExprNodeStack*, ExprNodeAddr); // MUST BE IMPLEMENTED BY
                                          // STUDENT
```

3. And, you are supposed to have worked on previous supportive functions to create polynomials as well as the expressions.

   Using the above (and other) definitions, functions, and setups, you should have implemented a function named as `createInfix()` and a conversion function named as **`convertInfixToPostfix()`** as described below.

   The function will have the prototype as follows,

```
ExprList createInfix(void);
void convertInfixToPostfix(ExprList* infixExpr,
                           ExprList* postfixExpr);
```

   A sample hint (from Wikipedia.org) –

   Example

   The infix expression "`5 + ((1 + 2) * 4) − 3`" can be written down like this in RPN:

```
5 1 2 + 4 * + 3 -
```

   The expression is evaluated left-to-right, with the inputs interpreted as shown in the following table (the *Stack* is the list of values the algorithm is "keeping track of" after the *Operation* given in the middle column has taken place):

Input	Operation	Stack	Comment
5	Push value	5	
1	Push value	1 5	
2	Push value	2 1 5	
+	Add	3 5	Pop two values (1, 2) and push result (3)

	Push value	4	
4		3	
		5	
*	Multiply	12	Pop two values (3, 4) and push result (12)
		5	
+	Add	17	Pop two values (5, 12) and push result (17)
3	Push value	3	
		17	
–	Subtract	14	Pop two values (17, 3) and push result (14)
	Result	(14)	

When a computation is finished, its result remains as the top (and only) value in the stack; in this case, 14.

4. Write a **menu** program to have the above options for the expressions.

   Your menu program should not use global data; data should be allowed to be read in and stored dynamically.

5. Name your program as **cis27Spring2014YourNameLab6Ex1.c**

   Make sure that the output is reasonable and detailed enough so that the user would understand the list – Use `printf()` measurably.

   Attach the output at the end of your source code (as comment).

```
*******************************
*         EXPRESSIONS         *
* 1. Creating/Updating infix *
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
*******************************
Select the option (1 through 6): 7

  You should not be in this class!


*******************************
*         EXPRESSIONS         *
* 1. Creating/Updating infix *
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
*******************************
Select the option (1 through 6): 1

// Appropriate details and display


*******************************
```

```
*         EXPRESSIONS        *
* 1. Creating/Updating infix *
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
******************************
Select the option (1 through 6): 2

// Appropriate details and display


******************************
*         EXPRESSIONS        *
* 1. Creating/Updating infix *
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
******************************
Select the option (1 through 6): 3

// Appropriate details and display


******************************
*         EXPRESSIONS        *
* 1. Creating/Updating infix *
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
******************************
Select the option (1 through 6): 4

// Appropriate details and display


******************************
*         EXPRESSIONS        *
* 1. Creating/Updating infix *
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
******************************
Select the option (1 through 6): 5

// Appropriate details and display


******************************
*         EXPRESSIONS        *
* 1. Creating/Updating infix *
```

```
* 2. Converting to postfix   *
* 3. Displaying infix        *
* 4. Displaying postfix      *
* 5. Evaluating expression   *
* 6. Quit                    *
******************************
Select the option (1 through 6): 6

// Appropriate details and display

  Having Fun!
```