CS 2250 **Data Structures and Problem Solving**
Assignment # 4 Due Date: April 10, 2007

Stack

Create a directory `${USER}.4` in your home. Keep all programs and datafiles for this assignment in this directory. Do each of the assigned programs in a separate directory within this directory. After you are done with the assignment, remove the executables, and execute the following commands on admiral:

```
% cd
% ~sanjiv/bin/handin cs2250 4
```

1. Design and implement a program calculator using stack operations as per following specifications:

   Function. This program will evaluate arithmetic expressions containing floating point numbers and the operators $+$, $-$, $*$, $/$, `%` (modulus), and ˆ (exponent).

   Input. The input is a series of arithmetic expressions entered interactively from the keyboard. The user is prompted to enter a fully and correctly parenthesized arithmetic expression made up of operators (the characters `'+'`, `'-'`, `'*'`, and `'/'`), parentheses, and floating point numbers. The end of the expression is marked by the expression terminator character, `'='`. There may be any number (including 0) of blanks between operators, parentheses, floating point numbers, and expression terminator. The outermost level of the expression does not need parentheses; that is, $(3 * (5 + 1))$ could also be expressed as $3 * (5 + 1)$.

   Floating point numbers must be expressed in decimal format: the whole part, followed by a decimal point (`'.'`), followed by the fractional part. The decimal point and fractional part are optional. Exponential notation (that is, 3.2E3) is not valid. The following are examples of floating point numbers processed by this program:

   $$5.0, 15.123, 0.0, 250, 27$$

   Examples of valid expressions (followed by the expression terminator, `'='`) are:

   ```
   (25 + 30.2) =
   100.0 - (5.3 * 12) =
   (2.78 + ( 53.44 - 3.3 ) ) * 0.5 =
   33 =
   ```

   Examples of invalid expressions are:

`1 + 2 + 3 =`	(Requires parentheses)
`3.2E3 * 5.0 =`	(Exponential notation not permitted)
`(1 + 2) + ((3 + 4) =`	(Parentheses do not match)
`3.55 * 0.5`	(Missing expression terminator)

The user terminates the program by entering the character `'#'` instead of an assignment statement.

Output. The user is prompted for each new expression to be evaluated with the following prompt:

```
Enter expression to evaluate or # to quit:
```

After the evaluation of each assignment statement, the results are printed to the screen:

`Result =` *value*

where *value* is a floating point number in decimal format, with a field width of 8 and a decimal field of 4.

Processing requirements are:

(a) This program must be able to be compiled and run, with minimal changes, on a variety of computer systems.

(b) This program must read floating point number inputs as characters and make the appropriate conversions. You can use the code given in the class to achieve the same.

Assumptions are:

(a) The expressions will be fully and correctly parenthesized, as described in the Input section.

(b) The expressions will be terminated by `'='`.

(c) The operations in expression will be valid at run time. This means that we will not try to divide by 0.

Use top-down design to design and code the implementation. Use good documentation. Submit the design and manual documentation along with the code on hard copy.