Programming Assignment #1

Program Postfix Evaluation

Function

The program evaluates postfix arithmetic expressions containing real numbers and the operators +, -, *, and /.

Input

The input is a series of arithmetic expressions entered interactively from the keyboard in postfix notation. The user is prompted to enter an expression made up of operators (the characters '+', '-', '*', and '/') and numbers. You can assume that the numbers are one digit numbers (0-9). They must be read in as characters (because the operators are read in as characters), and converted to numbers. The end of the expression is marked by the expression terminator character, '='. Operators and operands must be terminated/ separated by at least one blank.

The user terminates the program by entering the character "#" instead of a new expression.

Output:

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

Enter expression to evaluate = 4 5 - = <press enter key>

Result = *-1*

Try again [y/n]: y

If ‘y’ or ‘Y’, the screen will clear and display the prompt again.

Assumptions

1. The expressions are correctly entered in postfix notation.
2. Each expression is terminated by '='.
3. The operations in expressions are valid at run time. This means that we do not try to divide by zero.

Using a Stack for Evaluation

If we think about the postfix expressions, and how to evaluate them, we realize that when an operator is (the characters '+', '-', '*', and '/') is encountered, we apply it to the previous two operands (which have been read in, or are results of previous computation). For example, in the postfix expression:

6 2 / 5 + =

we read the 6 and the 2 and then read the operator. The operator is applied to the two numbers. If we place numbers, or results of equations on a stack, then we can apply operations to the two top elements of the stack. In the expression above, we read a 6, and push it onto the stack. Then we read a 2 and push it onto the stack. We read the character '/', and apply it to the two top elements of the stack. 6/2 is 3 so we push 3 onto the stack. We read a 5 and push it onto the stack. When we read the final '+', we apply it to the two top elements of the stack (3 and 5). Push the result to the stack. Upon reading the "=", the element in the stack is printed as the final answer.

Deliverables

1. Your source code in C or C++ (.c or .cpp) and its executable file (.exe)
2. A text file listing the expressions that you used to test your program and answers of those expressions.

Due date: 10 December 2011, 11.59pm