CS 343 – Structure of Programming Languages
Winter 2011

Programming Assignment #2
Due Date: Monday, February 21, 2011
Programming Language: C

Problem Specification

Write a program to perform the following two functions: 1) convert a given infix arithmetic expression to a postfix arithmetic expression and evaluate it, and 2) evaluate a given postfix arithmetic expression.

Assume a valid expression (infix or postfix) is entered by the user. Also, assume that an infix expression can contain only integers, binary operators (+, -, *, /, %, ^), '(', and ')'. Similarly, a postfix expression can contain only integers and binary operators. Each token (integers, binary operators, or parenthesis) in the expression is separated by at least one space.

The supplied `stack.h` and `stack.c` files provide functions that implement a stack using a linked list. You are required to use these functions in your implementation. Implement the functions declared in the `postfix.h` file. Define these functions in `postfix.c` file. Put the `main()` function of your program in `test.c` file.

Use the following algorithms in your program. Each algorithm uses a stack in support of its operations, and in each algorithm the stack is used for a different purpose.

The algorithm for converting an infix to postfix is as follows:

Read infix expression from left to right.
While there are more tokens in the infix expression, do the following:

- If the current token is an operand (integer), append it to postfix expression.
- If the current token is a left parenthesis, push it onto the stack.
- If the current token is an operator:
  - Pop operators (if there are any) from the stack with "stack precedence" equal or higher than the "input precedence" of the current token, and append popped operators to the postfix expression (see below for information on stack/input operator precedence).
  - Push the current token onto the stack.
- If the current token is a right parenthesis:

    - Pop operators from the stack and append them to postfix expression until a left parenthesis is at the top of the stack.
    - Pop (and discard) the left parenthesis from the stack.

Pop the remaining operators from the stack and append them to postfix expression.

The algorithm for evaluating a postfix expression is as follows:

Read postfix expression from left to right.
While there are more tokens in the postfix expression, do the following:

- If the current token is an operand (integer), push it onto the stack.
- If the current token is an operator:
    - Pop the top element into variable y.
    - Pop the next element into variable x.
    - Perform "x operator y".
    - Push the result of the calculator onto the stack.

Pop the top value of the stack. This is the result of evaluating the postfix expression.

Sample Execution

```
$ test
(1) Convert Infix to Postfix Expression
(2) Evaluate Postfix Expression
(3) Quit
Enter selection (1, 2, 3): 1
Enter Infix Expression: 3 * ( 4 - 2 ^ 5 ) + 6
    Postfix: 3 4 2 5 ^ - * 6 +
    Value: -78

Enter selection (1, 2, 3): 1
Enter Infix Expression: 3 ^ 2 ^ ( 1 + 2 )
    Postfix: 3 2 1 2 + ^ ^
    Value: 6561

Enter selection (1, 2, 3): 2
Enter Postfix Expression: 3 2 1 2 + ^ ^
    Value: 6561

Enter selection (1, 2, 3): 2
Enter Postfix Expression: 3 4 2 5 ^ - * 6 +
    Value: -78
```

```
Enter selection (1, 2, 3): 3
Bye.
```

Input and Stack Precedence of Operators

Operator	Input Precedence	Stack Precedence	Associativity
+ -	1	1	Left-to-Right
* / %	2	2	Left-to-Right
^	4	3	Right-to-Left
(	5	-1	

Deliverables

1. Upload your source files (*.c and *.h) on Blackboard.
    - I will use the submission date/time on Blackboard as your official submission date/time.
    - It is your responsibility to make sure the submission on Blackboard went through successfully.
2. Because of possible portability issues, make sure your program compiles on EOS machines before submitting. I will compile, run, and test your program on EOS when grading.
3. Access to the assignment will be turned off end of day on Thursday, February 24th (three days after the due date).
    - Late penalty (10% per day) applies if you submit after Monday, February 21st.