Programming Assignment 3 (Due Monday December 2, 2024)

Analysis of Algorithms

Expressions such as `2 + 3 * 5 - 9` are called *infix* expressions as the operators appear in between the associated operands, with appropriate parentheses present if the order of the operations needs to be specified. The same expression in *prefix* format reads `- + 2 * 3 5 9` , with the prefix format requiring that each operator appears before its associated operands. Similarly, the *postfix* expression of the same computation is `2 3 5 * + 9 -`. All three expressions share the same expression tree of the calculation.

In this project, you will

- build an expression tree, assuming the expression is initially given in the postfix format (your expression tree could be outputted using the simplest method you can think of, as long as it reveals the structure of your tree),
- do a traversal of the expression tree you have built and output the infix expression,
- do another traversal of your expression tree and output the prefix expression of the initially given postfix expression,
- and finally output the computation result by carrying out the prefix expression via your code.

What to Submit:

Please submit a write-up, and the source code to Blackboard before the due date of the assignment. In the write-up, please provide a concise description of your method and rationale for design and implementation decisions. Please also provide answers to the requirement instructions above, if any.

Your write-up may consist of three types of content: *Analysis*, *Results*, and *Source Code*, depending upon the individual assignment.

Analysis: This section may include relevant complexity and algorithm-related reasoning and related decisions necessary for your implementation or answering particular questions.

Results: You will be given a few testing cases as the input postfix expressions. Please output the requested expressions and values in your report, and explain the types of traversals you have used for these cases and any rationale behind your choices. Please show why your algorithm is correct (or why it is not :-)), and describe how depth-first search, breadth-first search, FIFO queue, or LIFO stack may be related with the output and evaluation of the three types of arithmetic expressions.

Source Code: The source code should be readable and commented appropriately. Internal comments should describe algorithms and variables, relating them to those described in your Analysis section. Briefly describe the inputs and outputs of your code. Your program should be ready to run on a machine for verification purposes if needed for addressing concerns from the instructor or the TA.

Bonus Points
You may make basic assumptions in your implementation of the project. For example, you may assume that the operands and operators are delineated by a white space, and only binary operators `+, -, *, /` will appear. Additionally, you may assume that the original input is valid. Bonus points will be given to programs that implement less-restrictive input conditions, as well as those that can print meaningful error messages when an input postfix expression is invalid.

Note: The programming assignments and associated write-ups must be done individually. However, discussing with your classmates and/or the instructor is encouraged.