CSC 228 Data Structures and Algorithms, Spring 2019
Instructor: Dr. Natarajan Meghanathan

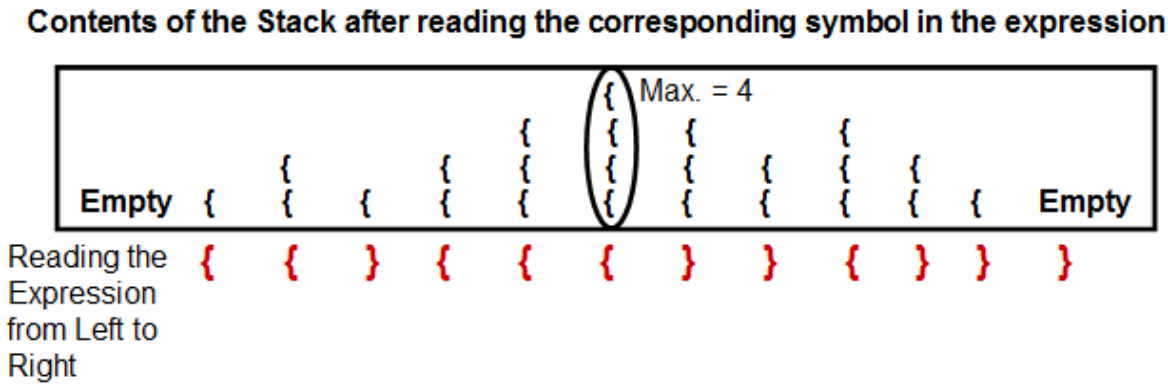
Project 3: Determination of Maximum Depth of Nested Parentheses in an Expression and Evaluation of an Expression in Pre-fix Format

Submission: Submit everything together as one PDF file in Canvas. Due: March 6th, 11.59 PM

Q1 - 65 pts) Consider the code given to you to determine whether an expression of parentheses is balanced or not using a Doubly Linked List implementation of a Stack. Modify the code (given in the main function) to determine the maximum depth of nested parentheses in a given expression, provided the expression is balanced.

Note that your code should be also able to check whether a given expression of parentheses is balanced or not. If an expression is not balanced, you should not print the value for the maximum depth of nested parentheses: your program should just print the message that the expression is not balanced and terminate.

For simplicity, you can assume that the only parenthesis symbols of use are { and }. The maximum depth of nested parentheses in a balanced expression is the largest number of open parentheses { in the stack at any time. For example, the maximum depth of nested parentheses in the expression { { } { { { } } { } } } is 4.



Run your modified code with the following expressions and determine the maximum depth of nested parentheses of each, if the expression is balanced. Include screenshot displaying the result.

a. { { } { { { } } { } } }
b. { { { } } { { { } } } }
c. { { { } { { } { { } } } { } } }
d. { { { } } { { } } }
e. { } { { } { } }

You need to submit the following for this question:
(1) Briefly describe your algorithm (along with a pseudo code) to determine the maximum depth of nested parentheses in a given expression. Analyze the time complexity of the algorithm.
(2) The complete code (including the modification/implementation of the algorithm in the main function)
(3) Screenshots of the execution of the algorithm/code for each of the above five inputs (a)-(e).

Q2 - 35 pts) You are given the code (including the main function) for evaluating an expression in post-fix format using Stack. Your task is to modify the code in the main function to input an expression in pre-fix format, reverse it (as discussed in class) and then evaluate the value of the reversed expression (scanned from left to right) using Stack.

You test your code with an expression string in pre-fix format that comprises of all the four operators (at least once) in a randomly chosen order with randomly chosen values in the range 1 to 9. For example, a sample expression string (pre-fix notation) could be input as **-, +, *, /, 5, 2, 3, 4, 8** for the expression (infix-notation) 5 / 2 * 3 + 4 - 8.

You need to submit the following for this question:
(1) The complete code (including the modification to evaluate an expression in pre-fix format) of the Stack class, Node class and the main function.
(2) Screenshot of the execution of the code for a sample pre-fix expression, like the one given above.