Exercises with Stacks

Exercise 1: from the book *The Algorithm Desing Manual.* Skiena, S.

A common problem for compilers and text editors is determining whether the parentheses in a string are balanced and properly nested. For example, the string '((())())()' contains properly nested pairs of parentheses while the strings ')()(' and '())' do not. Write a C program that returns *true* if a string contains properly nested and balanced parentheses, and *false* otherwise. Identify the position of the first offending parenthesis if the string is not properly nested and balanced.

Exercise 2:

Write a C program that reverses strings. For instance, given the string "*I am very good at data structures*", the algorithm must reverse it to "*serutcurts atad ta doog yrev ma I*". It is important to note that the program must change the given string, avoiding the creation of an additional variable to store the reversed string. The function must take linear time.

Exercise 3: from the book *Introduction to Algorithms.* Cormen T.H. *et al.*

Implement a stack in C using a singly linked list such that the operations *Push* and *Pop* still take constant time.

Exercise 4:

The Reverse Polish notation (RPN) is a prefixed notation used to describe arithmetic expressions. One of its advantages is the fact that it does not rely on the use of parenthesis to impose the evaluation order of the expression. Write a C program that accepts an arithmetic expression in RPN format and outputs the result of its evaluation. The only supported operations must be add (+), minus (-), divide (/) and multiply (*), and the operands are all positive and integer numbers composed by only one digit (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9). An example of a valid expression is '51+34-*', which evaluates to -6.

Exercise 5: from the book *Introduction to Algorithms.* Cormen T.H. *et al.*

Explain how to implement two stacks using only one array, e.g. A[1..*n*], in such a way that neither stack overflows unless the number of elements in both stacks sum to *n* +1. The *Push* and *Pop* operations, for both stacks, must run in *O*(1) (constant) time.