Stacks – Exercises

Ex 1

Consider the mathematical expression that includes several sets of nested parentheses, for example,

7 - ((x * ((x + y)) / (j-3) +y) / (4 – 2.5))

We want to ensure that the parentheses are nested correctly ; that is, we want to check that

- there are an equal number of right and left parentheses
- every right parenthesis is preceeded by a matching left parenthesis

Expressions such as ((a+b) or a+b(violate condition 1.

Expressions such as) a+b (-c or (a + b)) – (c + d violate condition 2.

1. Devise an algorithm for solving this problem, then write the corresponding C program (**with or without using stacks**)
2. What if expressions have different types of brackets () [] {} as in the following expression

   7 - { [x * [(x + y) / (j – 3)] + y] / (4 – 2.5) }

   Devise an algorithm for solving this problem, then write the corresponding C program.

Ex 2 – Calculator

Consider the following operation : ((a + b) * (c – (d -e)) ^ (f+g). This is called an infix expression as the binary operators are between operands.

To do the calculation, we a re going to split this into 2 distinct operations :

1. Transform the infix expression into a postfix expression. For instance :
   ab + c * de - - fg + ^

   The interesting part of this transformation is that we don't need parentheses anymore to express the precedence between operations !

   Design an algorithm that takes a string as an infix expression and transforms it into an postfix expression then write its corresponding C program.

2. Design an algorith to evaluate the postfix expression, then write its corresponding C program