Course: ESO207A – Data Structures and Algorithms

Indian Institute of Technology Kanpur

Programming Assignment 3 : *An elegant use of stack to evaluate an expression ?*

Most Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. Remember **- Before cheating the instructor, you are cheating yourself**. The onus of learning from a course lies first on you. So act wisely while working on this assignment.
- Refrain from collaborating with the students of other groups. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.
- This assignment has to be done in groups of 2 only. It is your responsibility to find a partner.
- In case of any issue related to this assignment, send an email at taruntk@cse.iitk.ac.in (and not to the instructor).

The Objective of the Assignment

In the lectures, we discussed an elegant application of stack to evaluate an arithmetic expression. You have to implement the same with slight changes (see below).

Tasks to be done

There are 2 tasks to be done.

1. Implementation of Stack

You need to provide an implementation of the stack data structure in C. All operations as discussed in the lecture have to be implemented. The items of the stack will be of type integers. In the second part of the problem, whenever you need a stack you **must** use the implementation of the stack developed in this part. You will get nil marks in this assignment if you deviate from this rule.

Note: In the second part of the problem, you must use the stack implemented here for storing characters as well, if required. (Recall that in C, integers and characters can be used interchangeably.)

2. Algorithm for expression evaluation

You need to implement the algorithm for expression evaluation we discussed in the class with the following change: $/$ and $-$ are right associative instead of left associative. Rest of the operators have the same associativity as discussed in class. For **clarification**: **+,*** are the only **left-associative** operators. You may assume that input expression is a valid arithmetic expression involving integers only. Also assume that expression contains operators $+, -, *, /$ and ˆ only and has no white spaces.
Note: This algorithm must work in linear time using the stack from part 1. No marks would be given for any brute force solution involving multiple passes of string.

Sample Input and output
The first line of input contains the length of the expression. The second line contains the expression.

1.
   ```
   Sample Input:
   9
   3+8/4/2-7
   Output:
   0
   ```

2.
   ```
   Sample Input:
   9
   5*2+8-7-4
   Output:
   15
   ```

3.
   ```
   Sample Input:
   23
   5*2+8*(107-(91-2^2)-23)
   Output:
   354
   ```

Some useful points to remember:

- This programming assignment has to be submitted on HackerRank – at the following link: Assignment 3.
- Please go through “a guide to HackerRank” available on the course website (under Section “Additional Supporting Material for the Course”). Maintain the same **naming convention** for the HackerRank account as mentioned in this guide.
- You will be graded on the basis of number of hidden test cases passed. Note that the test cases are hidden for you to analyze your code well. In case you are stuck, thinking about **corner cases** might help. For example: If your code gives a **TLE (Time Limit Exceeded error)**, it might be the case that your code is **inefficient** and might fail on **bigger inputs**. (For a general understanding, assume that an **O(n)** algorithm in **C** can very easily work under a time limit of **1 second** for **n** in order of 10^6).
- Some marks are reserved for good progamming practices - proper variable names, meaningful comments, indentation etc.