CSC/DSCI 2720: Data Structures Assignment 3

Due: 03/21/2023 @ 11:59 PM ET

Answer the below questions.
You may use whatever IDEs / editors you like, but you must submit your responses on iCollege as .py files. Failure to comply with this simple requirement will result in a score of Zero. Please, be careful not to be assigned a Zero score this way.

Few Rules to be followed, else will receive a score of ZERO

(1) Your submissions will work exactly as required.

(2) Your files shall not be incomplete or worse corrupted such that the file does not compile at all. Make sure you submit a file that compiles

(3) Your submission will show an output. Should you receive a Zero for no output shown do not bother to email me with "but the logic is perfect"!

Note that your program's output must exactly match the specs (design, style) given here for each problem to pass the instructor's test cases.
Design refers to how well your code is written (i.e., is it clear, efficient, and elegant), while Style refers to the readability of your code (commented, correct indentation, good variable names).

(Understanding how the algorithm works is very important. Any solution that directly copies and pastes from the lecture slides will lead to a zero score.)

1. (30 points) Suppose you are given a string containing only the characters '(', '[', '{', ')', ']', '}'. In this problem, you will write a function to determine whether the string has balanced parentheses.

Any solutions that always return true (or always return false) or otherwise try to game the distribution of test cases will receive zero credit.

Balanced parentheses means that every open parenthesis has exactly one close parenthesis corresponding to it and vice versa, and that for each pair the opening parenthesis precedes the closed parenthesis. The following strings have balanced parentheses:

()
([]{})
([{}])()

The following strings do not have balanced parentheses:

)(
([)
()(()]){()}

We consider the empty string to have balanced parentheses, as there is no imbalance. Your program should accept as input a single string containing only the characters '(', '[', '{', ')', ']', '}', and output a single line stating true or false. Your task is to complete the method called *hasBalancedParentheses()* that determines whether the string has balanced parentheses. At the end of the program add a comment mention the time and space complexity of your solution.

2. (40 points) Design a simple calculator that helps you solve the expression given. For example below are how the expressions are represented. These expression is also known as "post-fix" string expressions:

10 2 * 15 +

2 1 + 3 *

4 13 5 / +

The evaluations of the expressions are:

10 2 * 15 + → (10 * 2) + 15 = 35

2 1 + 3 * → (2 + 1) * 3 = 9

4 13 5 / + → 4 + (13 / 5) = 6

Input is an array of strings tokens, for example ["2","1","+","3","*"]. Your solution needs to think of this aspect.

Please be reminded that you need to design the calculator and not use in-built math methods from the programming language library to solve the expression. Doing So would lead to a straight score of Zero. At the end of the program add a comment mention the time and space complexity of your solution.

(3) (30 points) In this problem, you will write a program called *reverseLinkedList*() that reverses a singly linked list. A singly linked list is similar to the Java version of a linked list, which is called a doubly linked list, except that each node only maintains the location of the next node and not the previous node, and we only maintain the first node and not the last.

Your program should take as input a space-separated list of integers representing the original list, and output a space-separated list of integers representing the reversed list. Your algorithm must have a worst-case runtime in O(n) and a worst-case space complexity of O(1) beyond the input.

For example, if our input is:

5 7 1 2 3

then we should print:

3 2 1 7 5

Please note that you will need to create the following class as how we did in the lab:

- *MyLinkedListNode*: A class representing linked list nodes. Each node contains **an integer value**, and **a variable next** representing the next node in the list (which is null if we're at the last node). You may modify this class if you like.

Restrictions:

(a) In this problem, you must use the custom MyLinkedList class. You may not use the built-in LinkedList class. Any solutions that do not use MyLinkedList will receive a zero in all categories.

(b) Your algorithm must have a worst-case runtime in O(n) and a worst-case space complexity of O(1) beyond the input. Any solutions that are less efficient will receive at most half credit for all categories.

Very Very Important:

(1) Your code should be well commented which explains all the steps you are performing to solve the problem. A submission without code comments will immediately be deducted 15 points!

(2) As a comment in your code, please write your test-cases on how you would test your solution assumptions and hence your code. A submission without test cases will immediately be deducted 15 points! Example of cases to be tested for are like: What if the array input which is expected does not exist - that is, input is a null. How should your code handle such a situation? Maybe output some message like “Null input case, so no output”? What if the length of the array is one? … so on and so forth.

Please Remember: Although, written as comments - You will address your test cases in the form of code and not prose :)