Lab-10

IC 152

This lab session contains an assortment of questions from various topics, viz. recursion, string processing, data visualization, etc. The first three questions are to be attempted over the **HackerRank** platform. The rest, you may complete on your local machines and submit via moodle.

Questions

Q-1 Write a function which recursively performs binary search on a list of sorted integers. Binary search is a divide-and-conquer technique for searching for a given item in a list of sorted items. You will apply it to a list of integers. It works as follows. Let **IList** be the input list of integers which is sorted in ascending order. Let **X** be the integer which is to be searched in **IList**. In each iteration of binary search, we inspect the element in the list which is at the middle of **IList**. Call it **M**. There can be three possibilities: (1) if $\mathbf{M} = \mathbf{X}$, then we terminate and say that the element is found. (2) if $\mathbf{M} > \mathbf{X}$, then it is clear that **X**, if present in **IList**, can only be to the left of **M**. So, we recursively call binary search with the list of elements on the left of **M**. That is, the elements on the right of **M**, including **M**, are discarded. (3) if $\mathbf{M} < \mathbf{X}$ then the case is symmetric to case (2) and we recursively call binary search with the elements on the right of **M**.

If **X** is found in **IList**, then the function should return 1, else 0.

References for Binary search:

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-5-binary-search-trees-bst-sort/

https://en.wikipedia.org/wiki/Binary_search_algorithm

Input format:

search_element

lenth_of_list

list_contents (one per line)

(See test-cases for input format.)

Q-2 Given a sting of single-digit integers, binary operands and brackets/ parentheses, find out if the brackets/ parentheses are balanced. You may use the built-in stack data-structure of the Python language. In a stack, one can add and remove elements only from the top. There are referred to as push and pop operations. Hence, while performing a pop, the element which is retrieved is the one which was pushed the last. In other words, last-in-first-out.

A stack can be used for finding out if the parentheses are balanced in the input string, as follows. Each time an opening parenthesis is encountered in the input string, push it on the stack. Each time a closing parenthesis is encountered, pop an element from the stack and check if the opening and closing parentheses are of matching type. If not, then you know that the string is unbalanced. Integers and binary operands may be ignored. If at the end, there are extra opening/closing parenthesis, then again, you know that the string is unbalanced. Return 1 if the string is balanced and 0 otherwise.

E.g.

$1 * (2 + 3 * [4 - 5])$ should return 1.

$1 * (2 + 3 * [4 - 5\})$ should return 0.

$1 * (2 - 3$ should return 0.

A list data-structure in Python may be used as a stack. If Lst is a list, then Lst.append(X) will push the element X on the top of the stack and Lst.pop() will pop an element from the top of the stack.

Q-3 Given two strings S1 and S2 as input, check if they are anagrams of each other or not. S1 and S2 might very well be entire sentences. But they will not have more than 50 characters.

For example, if you are given following two strings:
Tom Marvolo Riddle

and
I am Lord Voldemort

The answer/output should be Yes. If the two input strings are not the anagram of each other, then the output should be No.

Q-4 Revisiting *pandas, fileIO, string processing* and *data visualization.*

1. Import the file "afi-top-100-quotes.csv" as a pandas dataframe.
2. Plot a histogram of the 'YEAR' column by considering bin size = 10 years.
3. Print the movies that have appeared more than once in the descending order of their number of occurrences.
4. Store the 'MOVIE' column in a separate variable called "movies" and then change the datatype of its elements to *string* [Hint: by using `astype("string")`.]
5. Split the movie titles using string `split()` and find the number of words for each of the titles and then plot a histogram of the number of the words by considering unit bin size.
6. From the split movie titles, find all the unique words and write them in a file along with their number of occurrences after sorting the words alphabetically. Please add the following header to the file

   Unique words   Number of occurrences
   ---

7. Repeat 4 - 7 for the 'QUOTE' column.

   Instructions for submitting Q-4: Create a pdf file by adding your codes and plots and upload to it to moodle (the same way you submitted for Lab-7)