Programming Assignment 3 (CS142)

Arrange strings in lexicographic order

In this assignment we will maintain a set of input strings in lexicographic order (https://en.wikipedia.org/wiki/Lexicographic_order). This ordering of strings is similar to how words are stored in a normal English dictionary. That is, given two strings "AB" and "AA" as input, "AA" comes before "AB" in lexicographic ordering.

The user will provide you with the number n of strings to be taken as input. You will store these strings in a linked list. Each time, you insert a new string into the linked list, you will figure out where to insert the new string to maintain the lexicographic ordering. For example, let $n = 2$. First, the user enters "cab". The linked list will have only one element corresponding to this string. Next, the user enters "abc". Since we maintain lexicographic order, we should enter the new element such that the list becomes "abc"->"cab". In order to find out where to insert a new string in the list, you need to figure out how to compare two strings in the lexicographic order. Luckily, this is very easy in Python. Let str1 and str2 denote two string variables. Then, if str1 comes before str2 in lexicographic ordering, the condition (str1<=str2) evaluates to TRUE.

Part 1: Take n as input from the user. Insert n strings given as input by the user into the linked list in the lexicographic order. Print the strings in the linked list.

Note: For the next part we will require all strings to have the same length. But, for this part, you should not impose any such restriction.

Implement a rudimentary spell checker

Let the linked list contain all the words in `some` dictionary. Now, given a new string (target string) by the user as input, we want to check if that string exists in the linked list. If it exists, we inform the user that the string is valid, else we find the string in the linked list that is closest to the target string and return the closest string. In order to find the closest string, we need a notion of distance between any two strings, and the closest string would be that string with the least distance to the target string. To make our lives easier we will assume all strings have equal length, say of length 5. Given two strings of length 5, "ABCDE" and

"UVCXY", we say the `Hamming` distance between these two strings is 4, since the strings differ in 4 places. Write a function to compute the Hamming distance between any two strings of the same length.

Part 2: Take as input n strings of length 5 from the user and a target string of length 5. Find the closest string in the linked list with respect to the target string and output the closest string. If there are choices for the closest string, output any of them arbitrarily.

Submission: Write both parts in a single python file. Name your submission as RollNo-Assgn3.py and upload on Google classroom.