CPE 202 Fall 2018

Lab2: Generating permutations in lexicographic order

Due: 10/5/18

Goal: Write a Python program to generate all the permutations of the characters in a string. This will give you a chance to review some simple Python constructs, i.e.. Strings and Lists and solidify your understanding of recursion.

Your program **<u>must</u>** meet the following specification. You are to write a Python function **perm_lex** that:

- Takes a string as a single input argument. You may assume the string consists of **distinct lower case** letters (in alphabetical order). You may assume the input is a string of letters <u>in alphabetical order</u>.
- Returns is a **<u>list</u>** of strings where each string represents a permutation of the input string. The list of permutations must be in lexicographic order. (This is basically the ordering that dictionaries use. Order by the first letter (alphabetically), if tie then use the second letter, etc.
- You need to use **design recipe** for this lab assignment. I will offer you the step 4, which is template and you need to cover all other steps.
- **NOTE**: You only allow to use basic library functions of Python! If the string is empty return empty list.

```
Argument:     abc
Returns:      [abc, acb, bac, bca, cab, cba]
```

Step 4: Templet (Pseudocode for a recursive algorithm to generate permutations in lexicographic order.) **You must follow this pseudo code.**

```
•  If the string contains a single character return a list containing that string

•  Loop through all character positions of the string containing the characters to be
   permuted, for each character:
      o  Form a simpler string by removing the character
      o  Generate all permutations of the simpler string recursively
      o  Add the removed character to the front of each permutation of the simpler
         word, and
      o  add the resulting permutation to a list

•  Return all these newly constructed permutations
```

Submit you Python function in a file called **perm_lex.py** to Polylearn and your test program in **perm_lex_testcases.py. Do not submit your files as zip file and use exactly the same file names and function name.**

Note: For a string with n characters, you program will return a list contain n! strings. Note that n! grows very quickly. For example : 15! is roughly $1.3*10^{12}$. Thus it is probably not a good idea to test your program with long strings.

Your test cases must cover all possible situations.