SLP_CODING_CHALLENGE_2—05/06/2020

QUESTION-1:

We are given 3 strings: str1, str2, and str3. Str3 is said to be a shuffle of str1 and str2 if it can be formed by interleaving the characters of str1 and str2 in a way that maintains the left to right ordering of the characters from each string.
For example, given str1="abc" and str2="def", str3="dabecf" is a valid shuffle since it preserves the character ordering of the two strings. So, given these 3 strings write a function that detects whether str3 is a valid shuffle of str1 and str2.

Solution:

```
def isShuffle(str1, str2, str3):
    if len(str1)+len(str2)!=len(str3):
        return False

    if not str1 or not str2 or not str3:
        if str1+str2==str3:
            return True
        else:
            return False

    if str1[0]!=str3[0] and str2[0]!=str3[0]:
        return False

    if str1[0]==str3[0] and isShuffle(str1[1:], str2, str3[1:]):
            return True
    if str2[0]==str3[0] and isShuffle(str1, str2[1:], str3[1:]):
            return True

    return False
```

****

QUESTION-2:

Write a function that accepts a single string input and returns the first non-repeated character.

Solution:

```
# Function to find the first non-repeating character in
def nonRepeatingChar(chars, n):
    dict = {}
    for index, char in enumerate(chars):
        frequency, prevIndex = dict.get(char, (0, index))
        dict[char] = (frequency + 1, index)
    min_index = n
    for key, values in dict.items():
        count, firstIndex = values
        if count == 1 and firstIndex < min_index:
            min_index = firstIndex
    return min_index
if __name__ == '__main__':
    str = "ABCDBAGHC"
    n = len(str)
    index = nonRepeatingChar(str, n)
    if index != n:
        print("The first non-repeating character in the is", str[index])
```

****