You can view this report online at : https://www.hackerrank.com/x/tests/1446752/candidates/51303625/report

Email:	hoonjung06@gmail.com
Test Name:	**ACSL 2022-23, Contest 3, Intermediate Division Programming Problem**
Taken On:	28 Mar 2023 16:20:44 PDT
Time Taken:	61 min 36 sec/ 4320 min
Invited by:	ACSL Contests
Invited on:	28 Mar 2023 16:00:18 PDT
Skills Score:	
Tags Score:	

100%

5/5

scored in **ACSL 2022-23, Contest 3, Intermediate Division Programming Problem** in 61 min 36 sec on 28 Mar 2023 16:20:44 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Tree Creation > Coding	1 hour 1 min 30 sec	5/ 5	✓

QUESTION 1

Correct Answer

Score 5

Tree Creation > Coding

QUESTION DESCRIPTION

PROBLEM STATEMENT:

Given a string to create 2 parallel arrays, process the letters in the string from left to right, inserting each letter into the first array in alphabetical order, one letter at a time. During this process, each letter is assigned an integer value in the second array as follows:

1. The first letter has a value of 0.
2. If the new letter is the first or last in the array, its value is one more than the value of the adjacent letter.
3. Otherwise, it is one more than the larger value of the two adjacent values. If the letter is already in the array, the new letter is placed before the existing letter.

Once the two arrays are created, print two different strings separated by a single space. For each letter in the array from left to right:

- Check if there is a letter to the left of it that has a value that is one greater than its value. Stop if you encounter any value that is less than its value. If a letter meets the condition above, add it to the first string.
- Check if there is a letter to the right of it that has a value that is one greater than its value. Stop if you encounter any value that is less than its value. If a letter meets the condition above, add it to the second string.
- Do not add the letter if it would be in both strings.

If either string is empty, print NONE instead.

EXAMPLE #1:

Here is how the array is built for the string PYTHONN:
The letter P illustrates rule 1; Y and H illustrate rule 2; the T, O and both Ns illustrate rule 3; and the final N illustrates what happens if there is a duplicate letter.

P	Letters	P								
	Values	0								
Y	Letters	P	Y							
	Values	0	1							
T	Letters	P	T	Y						
	Values	0	2	1						
H	Letters	H	P	T	Y					
	Values	1	0	2	1					
O	Letters	H	O	P	T	Y				
	Values	1	2	0	2	1				
N	Letters	H	N	O	P	T	Y			
	Values	1	3	2	0	2	1			
N	Letters	H	N	N	O	P	T	Y		
	Values	1	4	3	2	0	2	1		

In these final arrays, process each letter as follows:

- The letter H has a value of 1. Searching to its left, no letter is found. Searching to its right, a value of 2 is found. Add H to the second string.
- The 1st N has a value of 4. Searching to its left, there is no 5. Searching to its right, no 5 is found before encountering the 3. It is not added to either string.
- The 2nd N has a value of 3. Searching to its left, there is a 4. Searching to its right, there is no 4 before encountering the 2. Add N to the first string.
- The O has a value of 2. Searching to its left, there is a 3. Searching to its right, there is no 3 before encountering the 0. Add O to the first string.
- The P has a value of 0. Searching to its left, there is a 1. Searching to its right, there is a 1. Do not add it to either string.
- The T has a value of 2. Searching to its left, there is no 3 before encountering the 0. Searching to its right, there is no 3. It is not added to either string.
- The Y has a value of 1. Searching to its left, there is a 2. Searching to its right, no letter is found. It is added to the first string.

The output is NOY H.

EXAMPLE #2:

The string BINARYSEARCHTREE results in the following array:

Letters	A	A	B	C	E	E	E	H	I	N	R	R	R	S	T	Y
Values	2	1	0	3	5	4	2	3	1	2	5	4	3	5	6	4

Following the same process as above, the output is AERY CNS.

TASK:

Complete the function **onlyLeftOrRight** that is called from a program that inputs the following data as its parameters and outputs the following information for each individual input:

- The function has 1 parameter: a string, *input*, representing the string to use in building the two arrays
- The function should return a concatenated string of the first string and the second string separated by a single space

You may create additional functions that are called from **onlyLeftOrRight** if needed in solving the problem.

CONSTRAINTS:

The inputted string will be no more than 80 characters containing all capital letters.

DATA PROVIDED:

There are 5 sets of Sample Data for debugging and 5 sets of Test Data for scoring. You may create additional data sets for debugging your program.

CANDIDATE ANSWER

Language used: **Python 3**

```
#
# Complete the 'onlyLeftOrRight' function below.
#
# The function is expected to return a STRING.
# The function accepts STRING input as parameter.
#

def onlyLeftOrRight(input):
    letters=[]
    values=[]
    for l in range(len(input)):
        if l == 0:
            letters.append(input[l])
            values.append(0)
        else:
            for i in range(len(letters)):
                if (input[l] < letters[i]) or (input[l] == letters[i]):
                    letters.insert(i, input[l])
                    if i == 0:
                        values.insert(i, values[i]+1)
                    else:
                        values.insert(i, max(values[i-1], values[i])+1)
                    break
                elif i+1 == len(letters):
                    letters.append(input[l])
                    values.append(values[i]+1)
                    break
    left = ""
    right = ""
    for n in range(len(values)):
        l_true = 0
        r_true = 0
        kg = 1
        if (n != 0):
            for l in range(n):
                if values[l] == values[n]+1:
                    l_true = 1
                elif (l_true == 1) and (values[l] < values[n]):
                    l_true = 0

        kg = 1
        if (n+1 != len(values)):
            for j in range(len(values)-n-1):
                if values[j+1+n] < values[n]:
                    kg = 0
                if (kg == 1) and (values[j+1+n] == values[n]+1):
                    r_true = 1
```

```
49
50          if (r_true == 1) and (l_true == 0):
51              right += letters[n]
52          elif (r_true == 0) and (l_true == 1):
53              left += letters[n]
54
55      if len(left) == 0:
56          left = "NONE"
57      if len(right) == 0:
58          right = "NONE"
59      ans = left + " " + right
60      return ans
61
62
63
64
65
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	0	0.0737 sec	9.37 KB
Testcase 1	Easy	Sample case	Success	0	0.0458 sec	9.43 KB
Testcase 2	Medium	Sample case	Success	0	0.0835 sec	9.21 KB
Testcase 3	Medium	Sample case	Success	0	0.0552 sec	9.39 KB
Testcase 4	Hard	Sample case	Success	0	0.0726 sec	9.52 KB
Testcase 5	Easy	Hidden case	Success	1	0.052 sec	9.32 KB
Testcase 6	Medium	Hidden case	Success	1	0.0917 sec	9.36 KB
Testcase 7	Medium	Hidden case	Success	1	0.0964 sec	9.44 KB
Testcase 8	Hard	Hidden case	Success	1	0.0441 sec	9.4 KB
Testcase 9	Hard	Hidden case	Success	1	0.0455 sec	9.51 KB

No Comments