Rajalakshmi Engineering College

Name: Yugesh E T
Email: 240701613@rajalakshmi.edu.in
Roll no:
Phone: null
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 25

Section 1 : Coding

1. Problem Statement

Gina is working on a data analysis task where she needs to extract sublists from a given list of integers and find the median of each sublist. For each median found, she also needs to determine its negative index in the original list.

Help Gina by writing a program that performs these tasks.

Note: The median is the middle value in the sorted list of numbers, or the first value of the two middle values if the list has an even number of elements.

Example

Input

10

1 2 3 4 5 7 8 9 10 11

3

1 5

2 6

3 10

Output

3 : -8

4 : -7

7 : -5

Explanation

For the first range (1 to 5), the sublist is [1, 2, 3, 4, 5]. The median is 3, and its negative index in the original list is -8.

For the second range (2 to 6), the sublist is [2, 3, 4, 5, 7]. The median is 4, and its negative index in the original list is -7.

For the third range (3 to 10), the sublist is [3, 4, 5, 7, 8, 9, 10, 11]. The median is 7, and its negative index in the original list is -5.

Input Format

The first line of input consists of an integer N, representing the number of elements in the list.

The second line consists of N space-separated integers representing the elements of the list.

The third line consists of an integer R, representing the number of ranges.

The next R lines each consist of two integers separated by space representing the start and end indices (1-based) of the ranges.

Output Format

The output consists of n lines, displaying "X : Y" where X is the median of the

sublist and Y is the negative index in the original list.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10
1 2 3 4 5 7 8 9 10 11
3
1 5
2 6
3 10

Output: 3 : -8
4 : -7
7 : -5

Answer

```
N = int(input())
arr = list(map(int, input().split()))
R = int(input())

for _ in range(R):
    start, end = map(int, input().split())
    sublist = arr[start-1:end]
    sorted_sub = sorted(sublist)
    length = len(sorted_sub)

    if length % 2 == 1:
        median = sorted_sub[length // 2]
    else:
        median = sorted_sub[(length // 2) - 1]

    median_index = arr.index(median)

    print(f"{median} : {- (N - median_index)}")
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

You have two strings str1 and str2, both of equal length.

Write a Python program to concatenate the two strings such that the first character of str1 is followed by the first character of str2, the second character of str1 is followed by the second character of str2, and so on.

For example, if str1 is "abc" and str2 is "def", the output should be "adbecf".

Input Format

The input consists of two strings in each line.

Output Format

The output displays the concatenated string in the mentioned format.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: abc
def
Output: adbecf

Answer

```
str1 = input()
str2 = input()

result = ''.join([a + b for a, b in zip(str1, str2)])

print(result)
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Write a program to check if a given string is perfect.

A perfect string must satisfy the following conditions:

The string starts with a consonant.The string alternates between consonants and vowels.Each consonant appears exactly once.Vowels can occur consecutively multiple times but should not be followed immediately by a consonant.

If the string satisfies all these conditions, print “True”; otherwise, print “False”.

Input Format

The input consists of a string.

Output Format

The output prints "True" if the string is perfect. Otherwise, print "False".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: capacitor
Output: True

Answer

```
s = input()
vowels = 'aeiou'

if s[0] in vowels:
    print("False")
else:
    consonants_seen = set()
    consonants_seen.add(s[0])
    i = 1
    n = len(s)
    flag = True
    expect_vowels = True

    while i < n:
```

```
        if expect_vowels:
            if s[i] not in vowels:
                flag = False
                break
            start = i
            while i < n and s[i] in vowels:
                i += 1
            length_vowel_group = i - start
            if length_vowel_group == 1 and i < n and s[i] not in vowels:
                flag = False
                break
            expect_vowels = False
        else:
            if s[i] in vowels:
                flag = False
                break
            if s[i] in consonants_seen:
                flag = False
                break
            consonants_seen.add(s[i])
            i += 1
            expect_vowels = True

    print("True" if flag else "False")
```

Status : Partially correct **Marks : 5/10**