Rajalakshmi Engineering College

Name: King Paviyon Manova J
Email: 241501086@rajalakshmi.edu.in
Roll no: 241501086
Phone: 8903370369
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

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

```
def find_medians(lst, ranges):
    res = []
    for a, b in ranges:
        sub = sorted(lst[a-1:b])
        median = sub[(len(sub)-1)//2]
        idx = len(lst) - 1 - lst[::-1].index(median)
        res.append(f"{median} : {-len(lst) + idx}")
    return res

n = int(input())
lst = list(map(int, input().split()))
r = int(input())
ranges = [tuple(map(int, input().split())) for _ in range(r)]

for line in find_medians(lst, ranges):
    print(line)
```

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

2. 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

```
def perfect_string(s):
    vowels={'a','e','i','o','u'}
    consonants={chr(i) for i in range(ord('a'),ord('z') + 1)} - vowels
    if not s or s[0] not in consonants:
        return False
    seen_consonants=set()
    prev_char_type='consonant'
    for i in range(1,len(s)):
        char = s[i]
        if char in consonants:
            if char in seen_consonants:
                return False
            seen_consonants.add(char)
            if prev_char_type=='consonant':
```

```
            return False
        prev_char_type='consonant'
    elif char in vowels:
        if prev_char_type =='vowel':
            continue
        prev_char_type='vowel'
    else:
        return False
  return True
s=input().strip().lower()
if (perfect_string(s)):
  print("True")
else:
  print("False")
```

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

3. Problem Statement

Emily is a data analyst working for a company that collects feedback from customers in the form of text messages. As part of her data validation tasks, Emily needs to perform two operations on each message:

Calculate the sum of all the digits mentioned in the message.If the sum of the digits is greater than 9, check whether the sum forms a palindrome number.

Your task is to help Emily automate this process by writing a program that extracts all digits from a given message, calculates their sum, and checks if the sum is a palindrome if it is greater than 9.

Input Format

The input consists of a string s, representing the customer message, which may contain letters, digits, spaces, and other characters.

Output Format

The output prints an integer representing the sum of all digits in the string, followed by a space.

If the sum is greater than 9, print "Palindrome" if the sum is a palindrome, otherwise print "Not palindrome".

If the sum is less than or equal to 9, no palindrome check is required.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 12 books 4 pen
Output: 7

Answer

```
s = input()
temp = sum(int(c) for c in s if c.isdigit())
if temp <= 9:
    print(temp)
else:
    if str(temp) == str(temp)[::-1]:
        print(temp, "Palindrome")
    else:
        print(temp, "Not palindrome")
```

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