Rajalakshmi Engineering College

Name: Kaviya B J
Email: 240701246@rajalakshmi.edu.in
Roll no: 240701246
Phone: 9345986507
Branch: REC
Department: I CSE FC
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 : 30

Section 1 : Coding

1. Problem Statement

A company is creating email accounts for its new employees. They want to use a naming convention for email addresses that consists of the first letter of the employee's first name, followed by their last name, followed by @company.com.

The company also has a separate email domain for administrative employees.

Write a program that prompts the user for their first name, last name, role, and company and then generates their email address using the appropriate naming convention based on their role. This is demonstrated in the below examples.

Note:

The generated email address should consist of the first letter of the first name, the last name in lowercase, and a suffix based on the role and company, all in lowercase.

Input Format

The first line of input consists of the first name of an employee as a string.

The second line consists of the last name of an employee as a string.

The third line consists of the role of the employee as a string.

The last line consists of the company name as a string.

Output Format

The output consists of a single line containing the generated email address for the employee, following the specified naming convention.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: John
Smith
admin
iamNeo
Output: jsmith@admin.iamneo.com

Answer

```
s1=input()
s2=input()
s3=input()
s4=input()
if s3.lower()=="admin":
    email=(s1[0].lower() + s2.lower() + "@" + s3.lower() + "." + s4.lower() +
".com").replace(" ","")
    print(email)
else:
    email=(s1[0].lower() + s2.lower() + "@"  + s4.lower() + ".com").replace(" ","")
    print(email)
```

Status : Correct *Marks : 10/10*

2. 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

```
# You are using Python
n=int(input().strip())
```

```
lst=list(map(int,input().strip().split()))
r=int(input().strip())
for i in range(r):
    start,end=map(int,input().strip().split())
    sublist=lst[start-1:end]
    sublist_sorted=sorted(sublist)
    length=len(sublist_sorted)
    if length%2==1:
        median=sublist_sorted[length//2]
    else:
        median=sublist_sorted[(length//2)-1]
    index_in_original=lst.index(median)
    negative_index=index_in_original-len(lst)
    print(f"{median}: {negative_index}")
```

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

```
# You are using Python
s=input()
count=0
for char in s:
  if '0'<= char <='9':
     count += int(char)
print(count,end=" ")

if count>9:
   num=str(count)
   if num==num[::-1]:
      print("Palindrome")
   else:
      print("Not palindrome")
```

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