Rajalakshmi Engineering College

Name: visweswaran v
Email: 240701607@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 : 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

```
# You are using Python
# Input
first_name = input().strip()
last_name = input().strip()
role = input().strip()
company = input().strip()
email_prefix = first_name[0].lower() + last_name.lower()
if role.lower() == "admin":
    domain = f"admin.{company.lower()}.com"
else:
    domain = f"{company.lower()}.com"
```

```
email = f"{email_prefix}@{domain}"
print(email)
```

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

2. Problem Statement

Raja needs a program that helps him manage his shopping list efficiently. The program should allow him to perform the following operations:

Add Items: Raja should be able to add multiple items to his shopping list at once. He will input a space-separated list of items, each item being a string.

Remove Item: Raja should be able to remove a specific item from his shopping list. He will input the item he wants to remove, and if it exists in the list, it will be removed. If the item is not found, the program should notify him.

Update List: Raja might realize he forgot to add some items initially. After removing unnecessary items, he should be able to update his list by adding more items. Similar to the initial input, he will provide a space-separated list of new items.

Input Format

The first line consists of the initial list of integers should be entered as space-separated values.

The second line consists of the element to be removed should be entered as a single integer value.

The third line consists of the new elements to be appended should be entered as space-separated values.

Output Format

The output displays the current state of Raja's shopping list after each operation. After adding items, removing items, and updating the list, the program prints the updated shopping list in the following format:

"List1: [element1, element2, ... ,element_n]

List after removal: [element1, element2, ... ,element_n]

Final list: [element1, element2, ... ,element_n]".

If the item is not found in the removing item process, print the message "Element not found in the list".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3 4 5
3
6 7 8

Output: List1: [1, 2, 3, 4, 5]
List after removal: [1, 2, 4, 5]
Final list: [1, 2, 4, 5, 6, 7, 8]

Answer

```
# Input: initial shopping list as integers
initial_input = input()
shopping_list = list(map(int, initial_input.split()))
print("List1:", shopping_list)

# Input: item to remove (also as integer)
item_to_remove = int(input())

if item_to_remove in shopping_list:
    shopping_list.remove(item_to_remove)
    print("List after removal:", shopping_list)
else:
    print("Element not found in the list")
```

```
new_items = list(map(int, input().split()))
shopping_list.extend(new_items)

print("Final list:", shopping_list)
```

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

3. 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
def find_median_and_index(lst, ranges):
    for r in ranges:
        start, end = r
        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}")
n = int(input())
lst = list(map(int, input().split()))
r = int(input())
ranges = [tuple(map(int, input().split())) for _ in range(r)]

find_median_and_index(lst, ranges)
```

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