Rajalakshmi Engineering College

Name: Nishanthan V
Email: 241801190@rajalakshmi.edu.in
Roll no:
Phone: 8667548481
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 36.5

Section 1 : Coding

1. Problem Statement

Alex is tasked with managing the membership lists of several exclusive clubs. Each club has its own list of members, and Alex needs to determine the unique members who are part of exactly one club when considering all clubs together.

Your goal is to help Alex by writing a program that calculates the symmetric difference of membership lists from multiple clubs and then finds the total number of unique members.

Input Format

The first line of input consists of an integer k, representing the number of clubs.

The next k lines each contain a space-separated list of integers, where each

integer represents a member's ID.

Output Format

The first line of output displays the symmetric difference of the membership lists as a set.

The second line displays the sum of the elements in this symmetric difference.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
1 2 3
2 3 4
5 6 7
Output: {1, 4, 5, 6, 7}
23

Answer

```
k = int(input())

# Dictionary to count appearances
count = {}
for _ in range(k):
    members = list(map(int, input().split()))
    unique_members = []
    for m in members:
        if m not in unique_members:
            unique_members.append(m)
    for m in unique_members:
        if m in count:
            count[m] += 1
        else:
            count[m] = 1

# Members in exactly one club
result = []
for m in count:
    if count[m] == 1:
```

```
    result.append(m)

# Output as a set and the sum
print(set(result))
print(sum(result))
```

Status : Partially correct ***Marks : 6.5/10***

2. Problem Statement

Riley is analyzing DNA sequences and needs to determine which bases match at the same positions in two given DNA sequences. Each DNA sequence is represented as a tuple of integers, where each integer corresponds to a DNA base.

Your task is to write a program that compares these two sequences and identifies the bases that match at the same positions and print it.

Input Format

The first line of input consists of an integer n, representing the size of the first tuple.

The second line contains n space-separated integers, representing the elements of the first DNA sequence tuple.

The third line of input consists of an integer m, representing the size of the second tuple.

The fourth line contains m space-separated integers, representing the elements of the second DNA sequence tuple.

Output Format

The output is a space-separated integer of the matching bases at the same positions in both sequences.

Refer to the sample output for format specifications.

Sample Test Case

Input: 4
5 1 8 4
4
4 1 8 2

Output: 1 8

Answer

```
# Read size and elements of first DNA sequence
n = int(input())
seq1 = list(map(int, input().split()))

# Read size and elements of second DNA sequence
m = int(input())
seq2 = list(map(int, input().split()))

# Compare only up to the length of the shorter sequence
min_len = min(n, m)

# Collect matching bases at the same positions
matches = []
for i in range(min_len):
    if seq1[i] == seq2[i]:
        matches.append(str(seq1[i]))

# Output the result
print(" ".join(matches))
```

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

3. Problem Statement

Emily is a librarian who keeps track of books borrowed and returned by her patrons. She maintains four sets of book IDs: the first set represents books borrowed, the second set represents books returned, the third set represents books added to the collection, and the fourth set represents books that are now missing. Emily wants to determine which books are still borrowed but not returned, as well as those that were added but are now missing. Finally, she needs to find all unique book IDs from both results.

Help Emily by writing a program that performs the following operations on four sets of integers:

Compute the difference between the borrowed books (first set) and the returned books (second set).Compute the difference between the added books (third set) and the missing books (fourth set).Find the union of the results from the previous two steps, and sort the final result in descending order.

Input Format

The first line of input consists of a list of integers representing borrowed books.

The second line of input consists of a list of integers representing returned books.

The third line of input consists of a list of integers representing added books.

The fourth line of input consists of a list of integers representing missing books.

Output Format

The first line of output displays the difference between sets P and Q, sorted in descending order.

The second line of output displays the difference between sets R and S, sorted in descending order.

The third line of output displays the union of the differences from the previous two steps, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3
2 3 4
5 6 7
6 7 8
Output: [1]
[5]

[5, 1]

Answer

```
# You are using Python
# Read input sets
P = set(map(int, input().split()))  # Borrowed books
Q = set(map(int, input().split()))  # Returned books
R = set(map(int, input().split()))  # Added books
S = set(map(int, input().split()))  # Missing books

# Step 1: Borrowed but not returned
diff1 = sorted(P - Q, reverse=True)
print(diff1)

# Step 2: Added but now missing
diff2 = sorted(R - S, reverse=True)
print(diff2)

# Step 3: Union of both differences
final_union = sorted(set(diff1).union(diff2), reverse=True)
print(final_union)
```

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

4. Problem Statement

Riya owns a store and keeps track of item prices from two different suppliers using two separate dictionaries. He wants to compare these prices to identify any differences. Your task is to write a program that calculates the absolute difference in prices for items that are present in both dictionaries. For items that are unique to one dictionary (i.e., not present in the other), include them in the output dictionary with their original prices.

Help Riya to implement the above task using a dictionary.

Input Format

The first line of input consists of an integer n1, representing the number of items in the first dictionary.

The next n1 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

The following line consists of an integer n2, representing the number of items in the second dictionary

The next n2 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

Output Format

The output should display a dictionary that includes:

1. For items common to both dictionaries, the absolute difference between their prices.
2. For items that are unique to one dictionary, the original price from that dictionary.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
4
4
1
8
7

Output: {4: 4, 8: 7}

Answer

```
# Read first dictionary
n1 = int(input())
dict1 = {}
order1 = []
for _ in range(n1):
    key = int(input())
```

```
    value = int(input())
    dict1[key] = value
    order1.append(key)

# Read second dictionary
n2 = int(input())
dict2 = {}
order2 = []
for _ in range(n2):
    key = int(input())
    value = int(input())
    dict2[key] = value
    order2.append(key)

# Create result dictionary in correct order
result = {}

# First, process keys from dict1 in their input order
for key in order1:
    if key in dict2:
        result[key] = abs(dict1[key] - dict2[key])
    else:
        result[key] = dict1[key]

# Then, add keys unique to dict2 in their input order
for key in order2:
    if key not in dict1:
        result[key] = dict2[key]

# Output result
print(result)
```

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