Rajalakshmi Engineering College

Name: Aishwarya R
Email: 241501011@rajalakshmi.edu.in
Roll no: 241501011
Phone: null
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 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 37.5

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

```
def symmetric_difference_of_multiple_sets():
    k = int(input())

    first_club_members_str = input().split()
    current_symmetric_diff = set(map(int, first_club_members_str))

    for _ in range(k - 1):
        club_members_str = input().split()
        next_club_members = set(map(int, club_members_str))
        current_symmetric_diff = 
current_symmetric_diff.symmetric_difference(next_club_members)

    sorted_symmetric_diff = sorted(list(current_symmetric_diff))

    print(f"{{{', '.join(map(str, sorted_symmetric_diff))}}}")
    print(sum(sorted_symmetric_diff))

symmetric_difference_of_multiple_sets()
```

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

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

```
n = int(input())
seq1 = tuple(map(int, input().split()))

m = int(input())
seq2 = tuple(map(int, input().split()))

min_len = min(n, m)

matches = []
for i in range(min_len):
   if seq1[i] == seq2[i]:
       matches.append(seq1[i])

print(*matches)
```

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

3. Problem Statement

James is an engineer working on designing a new rocket propulsion system. He needs to solve a quadratic equation to determine the optimal launch trajectory. The equation is of the form ax2 +bx+c=0.

Your task is to help James find the roots of this quadratic equation. Depending on the discriminant, the roots might be real and distinct, real and equal, or complex. Implement a program to determine and display the roots of the equation based on the given coefficients.

Input Format

The first line of input consists of an integer N, representing the number of coefficients.

The second line contains three space-separated integers a,b, and c representing the coefficients of the quadratic equation.

Output Format

The output displays:

1. If the discriminant is positive, display the two real roots.
2. If the discriminant is zero, display the repeated real root.

3. If the discriminant is negative, display the complex roots as a tuple with real and imaginary parts.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 5 6

Output: (-2.0, -3.0)

Answer

```
def solve_quadratic_equation_without_cmath():
    N = int(input())
    a, b, c = map(int, input().split())

    discriminant = (b**2) - (4*a*c)

    if discriminant >= 0:
        if discriminant == 0:
            root = -b / (2*a)
            print(f"({root},)")
        else:


            root_plus = (-b + discriminant**0.5) / (2*a)
            root_minus = (-b - discriminant**0.5) / (2*a)
            print(f"({root_plus}, {root_minus})")
    else:
        real_part = -b / (2*a)
        imaginary_part = (abs(discriminant)**0.5) / (2*a)

        print(f"(({real_part}, {imaginary_part}), ({real_part}, {-imaginary_part}))")

solve_quadratic_equation_without_cmath()
```

Status : Partially correct ***Marks : 7.5/10***

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

```
def compare_prices():
    n1 = int(input())
    dict1_items_list = []
    for _ in range(n1):
        key = int(input())
        value = int(input())
        dict1_items_list.append((key, value))

    dict1 = dict(dict1_items_list)

    n2 = int(input())
    dict2_items_list = []
    for _ in range(n2):
        key = int(input())
        value = int(input())
        dict2_items_list.append((key, value))

    dict2 = dict(dict2_items_list)
    result_dict = {}

    for key, value in dict1_items_list:
        if key in dict2:
            result_dict[key] = abs(value - dict2[key])
        else:
            result_dict[key] = value

    for key, value in dict2_items_list:
        if key not in dict1:
            result_dict[key] = value

    print(result_dict)

compare_prices()
```

Status : Correct *Marks : 10/10*