Rajalakshmi Engineering College

Name: hemachandiran A
Email: 240701188@rajalakshmi.edu.in
Roll no: 240701188
Phone: 9655742740
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

240701188

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

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 = {}
    keys_order = []
```

240701188

```
    for _ in range(n1):
        key = int(input())
        value = int(input())
        dict1[key] = value
        keys_order.append(key)  # Store insertion order

    n2 = int(input())
    dict2 = {}

    for _ in range(n2):
        key = int(input())
        value = int(input())
        dict2[key] = value
        if key not in keys_order:
            keys_order.append(key)  # Maintain order of appearance

    output_dict = {}

    for key in keys_order:
        if key in dict1 and key in dict2:
            output_dict[key] = abs(dict1[key] - dict2[key])
        elif key in dict1:
            output_dict[key] = dict1[key]
        else:
            output_dict[key] = dict2[key]

    print(output_dict)

compare_prices()
```

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

2. Problem Statement

Alex is working with grayscale pixel intensities from an old photo that has been scanned in a single row. To detect edges in the image, Alex needs to calculate the differences between each pair of consecutive pixel intensities.

Your task is to write a program that performs this calculation and returns

240701188

the result as a tuple of differences.

Input Format

The first line of input contains an integer n, representing the number of pixel intensities.

The second line contains n space-separated integers representing the pixel intensities.

Output Format

The output displays a tuple containing the absolute differences between consecutive pixel intensities.

Refer to the sample output for format specifications.

Sample Test Case

Input: 5
200 100 20 80 10

Output: (100, 80, 60, 70)

Answer

```
# You are using Python
def calculate_differences():
    n = int(input())
    pixel_intensities = list(map(int, input().split()))

    differences = tuple(abs(pixel_intensities[i] - pixel_intensities[i + 1]) for i in
range(n - 1))

    print(differences)

calculate_differences()
```

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

3. Problem Statement

240701188

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

```
# You are using Python
def find_matching_bases():
```

240701188

```
    n = int(input())
    sequence1 = tuple(map(int, input().split()))

    m = int(input())
    sequence2 = tuple(map(int, input().split()))

    matching_bases = [sequence1[i] for i in range(min(n, m)) if sequence1[i] ==
sequence2[i]]

    print(" ".join(map(str, matching_bases)))

find_matching_bases()
```

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

4. 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.

240701188

240701188

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 5 6

Output: (-2.0, -3.0)

Answer

```
# You are using Python
import math

def find_quadratic_roots():
    n = int(input())
    if n != 3:
        return

    a, b, c = map(int, input().split())
    discriminant = b**2 - 4*a*c

    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2*a)
        root2 = (-b - math.sqrt(discriminant)) / (2*a)
        print((root1, root2))
    elif discriminant == 0:
        root = -b / (2*a)
        print((root,))
    else:
        real_part = -b / (2*a)
        imaginary_part = math.sqrt(abs(discriminant)) / (2*a)
        print(((real_part, imaginary_part), (real_part, -imaginary_part)))

find_quadratic_roots()
```

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