Rajalakshmi Engineering College

Name: Kaviya AK
Email: 240701245@rajalakshmi.edu.in
Roll no: 2116240701245
Phone: 6381101960
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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

Noah, a global analyst at a demographic research firm, has been tasked with identifying which country experienced the largest population growth over a two-year period. He has a dataset where each entry consists of a country code and its population figures for two consecutive years. Noah needs to determine which country had the highest increase in population and present the result in a specific format.

Help Noah by writing a program that outputs the country code with the largest population increase, along with the increase itself.

Input Format

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

Each of the following N blocks contains three lines:

1. The first line is a country code.
2. The second line is an integer representing the population of the country in the first year.
3. The third line is an integer representing the population of the country in the second year.

Output Format

The output displays the country code and the population increase in the format {code: difference}, where code is the country code and difference is the increase in population.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
01
1000
1500
02
2000
2430
03
1500
3000
Output: {03:1500}

Answer

```
def find_largest_population_increase(n, data):
    max_increase = 0
    country_code = ""
    for i in range(n):
        code = data[i][0]
        population_first_year = data[i][1]
        population_second_year = data[i][2]

        increase = population_second_year - population_first_year
        if increase > max_increase:
            max_increase = increase
            country_code = code

    return country_code, max_increase


N = int(input())
data = []

for _ in range(N):
    country_code = input().strip()
    population_first_year = int(input())
    population_second_year = int(input())
    data.append((country_code, population_first_year, population_second_year))
code, increase = find_largest_population_increase(N, data)

print(f"{{{code}:{increase}}}")
```

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

2. 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

```
# You are using Python
import math

def find_roots(a, b, c):
    discriminant = b**2 - 4*a*c

    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2 * a)
        root2 = (-b - math.sqrt(discriminant)) / (2 * a)
        return (root1, root2)

    elif discriminant == 0:
        root = -b / (2 * a)
        return (root,)

    else:
        real_part = -b / (2 * a)
        imaginary_part = math.sqrt(-discriminant) / (2 * a)
        return ((real_part, imaginary_part), (real_part, -imaginary_part))

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

```
roots = find_roots(a, b, c)

if len(roots) == 1:
    print(f"({roots[0]})")
else:
    if isinstance(roots[0], tuple):
        print(f"({roots[0]}, {roots[1]})")
    else:
        print(f"({roots[0]}, {roots[1]})")
```

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

3. 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

```
# You are using Python
n = int(input())
seq1 = tuple(map(int, input().split()))

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

min_length = min(n, m)

matching_bases = []

for i in range(min_length):
    if seq1[i] == seq2[i]:
        matching_bases.append(seq1[i])
print(" ".join(map(str, matching_bases)))
```

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

4. 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 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
n = int(input())
pixels = list(map(int, input().split()))
differences = tuple(abs(pixels[i] - pixels[i+1]) for i in range(n - 1))
print(differences)
```

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