

Rajalakshmi Engineering College

Name: madhavan R
Email: 240701294@rajalakshmi.edu.in
Roll no: 240701294
Phone: 6381021871
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

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

```
def main():  
    borrowed_books = set(map(int, input().split()))
```

```
returned_books = set(map(int, input().split()))
added_books = set(map(int, input().split()))
missing_books = set(map(int, input().split()))

still_borrowed = borrowed_books - returned_books
missing_added = added_books - missing_books

result1 = sorted(still_borrowed, reverse=True)
result2 = sorted(missing_added, reverse=True)
final_result = sorted(result1 + result2, reverse=True)

print(result1)
print(result2)
print(final_result)

if __name__ == "__main__":
    main()
```

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

```
def find_matching_bases():
    n = int(input().strip())
    sequence1 = tuple(map(int, input().strip().split()))
    m = int(input().strip())
    sequence2 = tuple(map(int, input().strip().split()))
    min_length = min(n, m)
    matching_bases = []
    for i in range(min_length):
        if sequence1[i] == sequence2[i]:
            matching_bases.append(sequence1[i])
    print(" ".join(map(str, matching_bases)))
find_matching_bases()
```

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 $ax^2 + bx + c = 0$.

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

```
import cmath
```

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

```
    if discriminant > 0:
```

```
        root1 = (-b + discriminant**0.5) / (2*a)
```

```
        root2 = (-b - discriminant**0.5) / (2*a)
```

```
        return (root1, root2)
```

```

elif discriminant == 0:
    root = -b / (2*a)
    return (root,)
else:
    real_part = -b / (2*a)
    imaginary_part = (abs(discriminant)**0.5) / (2*a)
    root1 = (real_part, imaginary_part)
    root2 = (real_part, -imaginary_part)
    return (root1, root2)
N = int(input())
coefficients = list(map(int, input().split()))
a, b, c = coefficients
roots = find_roots(a, b, c)
if len(roots) == 1:
    print(roots[0])
else:
    print(roots)

```

Status : Partially correct

Marks : 7.5/10

4. 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

```
# You are using Python
```

```
N = int(input())
```

```
max_increase = 0
```

```
country_code = ""
```

```
for _ in range(N):
```

```
    code = input().strip()
```

```
    population_year1 = int(input().strip())
```

```
    population_year2 = int(input().strip())
```

```
    increase = population_year2 - population_year1
```

```
    if increase > max_increase:
```

```
        max_increase = increase
```

```
country_code = code  
print(f"{{{country_code}:{max_increase}}}")
```

Status : Correct

Marks : 10/10