Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
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

Samantha is working on a text analysis tool that compares two words to find common and unique letters. She wants a program that reads two words, w1, and w2, and performs the following operations:

Print the letters common to both words, in alphabetical order.Print the letters that are unique to each word, in alphabetical order.Determine if the set of letters in the first word is a superset of the letters in the second word.Check if there are no common letters between the two words and print the result as a Boolean value.

Ensure the program ignores case differences and leading/trailing spaces in the input words.

Your task is to help Samantha in implementing the same.

Input Format

The first line of input consists of a string representing the first word, w1.

The second line consists of a string representing the second word, w2.

Output Format

The first line of output should display the sorted letters common to both words, printed as a list.

The second line should display the sorted letters that are unique to each word, printed as a list.

The third line should display a Boolean value indicating if the set of letters in w1 is a superset of the set of letters in w2.

The fourth line should display a Boolean value indicating if there are no common letters between w1 and w2.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: program
Peace
Output: ['a', 'p']
['c', 'e', 'g', 'm', 'o', 'r']
False
False

Answer

```
# You are using Python
def analyze_words(w1, w2):
    w1 = w1.strip().lower()
    w2 = w2.strip().lower()
    set1 = set(w1)
    set2 = set(w2)
    common_letters = sorted(set1.intersection(set2))
    unique_letters = sorted(set1.symmetric_difference(set2))
    is_superset = set1.issuperset(set2)
```

```
    no_common = len(common_letters) == 0
    return common_letters, unique_letters, is_superset, no_common

if __name__ == "__main__":
    w1 = input()
    w2 = input()
    common, unique, superset, no_common = analyze_words(w1, w2)
    print(common)
    print(unique)
    print(superset)
    print(no_common)
```

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

```
# You are using Python
def find_matching_bases(seq1, seq2):
    matches = []
    for a, b in zip(seq1, seq2):
        if a == b:
            matches.append(a)
    return matches

if __name__ == "__main__":
    n = int(input())
    seq1 = tuple(map(int, input().split()))
    m = int(input())
    seq2 = tuple(map(int, input().split()))

    matching_bases = find_matching_bases(seq1, seq2)
    print(" ".join(map(str, 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 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 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)
        return ((real_part, imaginary_part), (real_part, -imaginary_part))

if __name__ == "__main__":
    N = int(input())
    coefficients = list(map(int, input().split()))
    a, b, c = coefficients[0], coefficients[1], coefficients[2]

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

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

4. 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
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
    still_added = added_books - missing_books

    still_borrowed_sorted = sorted(still_borrowed, reverse=True)
    still_added_sorted = sorted(still_added, reverse=True)
```

```
    unique_books = sorted(still_borrowed.union(still_added), reverse=True)

    print(still_borrowed_sorted)
    print(still_added_sorted)
    print(unique_books)

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

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