Rajalakshmi Engineering College

Name: Manju sri N
Email: 241801151@rajalakshmi.edu.in
Roll no: 241801151
Phone: 8946059431
Branch: REC
Department: I AI & DS FC
Batch: 2028
Degree: B.E - AI & DS

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

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

```
# Function to compute the desired dictionary
def compare_prices(n1, dict1, n2, dict2):
    result = {}

    # Iterate over items in the first dictionary
```

```
    for item, price in dict1.items():
        if item in dict2:
            # If the item exists in both dictionaries, calculate the absolute difference
            result[item] = abs(price - dict2[item])
        else:
            # If the item is only in the first dictionary, add it as is
            result[item] = price

    # Iterate over items in the second dictionary
    for item, price in dict2.items():
        if item not in dict1:
            # If the item is only in the second dictionary, add it as is
            result[item] = price

    return result

# Read input values
n1 = int(input())  # Number of items in the first dictionary
dict1 = {}
for _ in range(n1):
    item = int(input())  # item key
    price = int(input())  # item price
    dict1[item] = price

n2 = int(input())  # Number of items in the second dictionary
dict2 = {}
for _ in range(n2):
    item = int(input())  # item key
    price = int(input())  # item price
    dict2[item] = price

# Get the result after comparing prices
result = compare_prices(n1, dict1, n2, dict2)

# Print the resulting dictionary
print(result)
```

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

2. 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

```
# You are using Python
# Function to compute the symmetric difference of multiple sets
def compute_symmetric_difference(sets):
    result = sets[0]  # Start with the first club's members
    for club_set in sets[1:]:  # Iterate over the remaining clubs
```

```
        result ^= club_set  # Perform symmetric difference using ^= operator
    return result

# Read the input
k = int(input())  # Number of clubs
sets = []

# Read each club's member list and convert it into a set
for _ in range(k):
    club_members = set(map(int, input().split()))  # Convert the space-separated
list into a set of integers
    sets.append(club_members)

# Compute the symmetric difference of all sets
result_set = compute_symmetric_difference(sets)

# Output the result
print(result_set)  # The symmetric difference as a set
print(sum(result_set))  # The sum of elements in the result set
```

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  # For handling complex square roots

# Function to calculate the roots of the quadratic equation
def solve_quadratic(a, b, c):
    # Calculate the discriminant
    D = b**2 - 4*a*c

    # If D > 0, two real roots
    if D > 0:
        root1 = (-b + cmath.sqrt(D)) / (2 * a)
        root2 = (-b - cmath.sqrt(D)) / (2 * a)
        # Display the real parts of the roots
        return (root1.real, root2.real)

    # If D == 0, one real root (repeated)
    elif D == 0:
        root = -b / (2 * a)
        return (root,)

    # If D < 0, complex roots
    else:
```

```
    root1 = (-b + cmath.sqrt(D)) / (2 * a)
    root2 = (-b - cmath.sqrt(D)) / (2 * a)
    return ((root1.real, root1.imag), (root2.real, root2.imag))

# Read the input
N = int(input())  # Number of coefficients, which is always 3 in this case
a, b, c = map(int, input().split())  # Coefficients of the quadratic equation

# Solve the quadratic equation and print the roots
roots = solve_quadratic(a, b, c)

# Output the roots
if len(roots) == 1:
  print(f"({roots[0]})")
else:
  print(f"({roots[0]}, {roots[1]})")
```

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

4. 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):
    # Remove leading/trailing spaces and convert to lowercase
    w1 = w1.strip().lower()
    w2 = w2.strip().lower()

    # Convert to sets
    set_w1 = set(w1)
    set_w2 = set(w2)

    # Find common letters (intersection) and sort them
    common_letters = sorted(set_w1 & set_w2)

    # Find unique letters in each word (symmetric difference) and sort them
```

```
    unique_letters = sorted((set_w1 - set_w2) | (set_w2 - set_w1))

    # Check if set_w1 is a superset of set_w2
    is_superset = set_w1 >= set_w2

    # Check if there are no common letters (empty intersection)
    no_common = len(common_letters) == 0

    # Output the results
    print(common_letters)
    print(unique_letters)
    print(is_superset)
    print(no_common)

# Input reading
w1 = input().strip()
w2 = input().strip()

# Call the function to analyze the words
analyze_words(w1, w2)
```

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