Started on	Friday, 24 May 2024, 8:09 AM
State	Finished
Completed on	Friday, 24 May 2024, 9:00 AM
Time taken	50 mins 43 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

Given a tuple and a positive integer k, the task is to find the count of distinct pairs in the tuple whose sum is equal to **K**.

Examples:

Input: t = (5, 6, 5, 7, 7, 8), K = 13
Output: 2
Explanation:
Pairs with sum K(= 13) are {(5, 8), (6, 7), (6, 7)}.
Therefore, distinct pairs with sum K(= 13) are { (5, 8), (6, 7) }.
Therefore, the required output is 2.

For example:

Input	Result
1,2,1,2,5 3	1
1,2 0	0

Answer: (penalty regime: 0 %)

```
def count_distinct_pairs(t, K):
    # Convert the tuple to a set to remove duplicates
    t = set(t)
    t = list(t)
    for i in range(0, len(t)):
        t[i] = int(t[i])
    # Initialize the count of distinct pairs
    count = 0

    # Iterate through the set of elements
    for num in t:
        # Check if the complement (K - num) is also in the set
        if (K - num) in t:
            # Increment the count of distinct pairs
            count += 1

    # Divide the count by 2 to get the number of distinct pairs
    return count // 2

# Example usage
arr1 = list(map(int, input().split(',')))
K1 = int(input())
print(count_distinct_pairs(arr1, K1))

```

	Input	Expected	Got	
✓	5,6,5,7,7,8 13	2	2	✓
✓	1,2,1,2,5 3	1	1	✓

	Input	Expected	Got	
✓	1,2 0	0	0	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeating elements and the total number of such non-repeating elements.

Input Format:

The first line contains space-separated values, denoting the size of the two arrays in integer format respectively.

The next two lines contain the space-separated integer arrays to be compared.

Sample Input:

5 4

1 2 8 6 5

2 6 8 10

Sample Output:

1 5 10

3

Sample Input:

5 5

1 2 3 4 5

1 2 3 4 5

Sample Output:

NO SUCH ELEMENTS

For example:

Input	Result
5 4 1 2 8 6 5 2 6 8 10	1 5 10 3

Answer: (penalty regime: 0 %)

```
def find_non_repeating_elements(arr1, arr2):
    # Convert arrays to sets to remove duplicates
    set1 = set(arr1)
    set2 = set(arr2)

    # Find the intersection of the two sets
    common_elements = set1.intersection(set2)

    # Find the non-repeating elements by subtracting the common elements from each set
    non_repeating_elements1 = set1 - common_elements
    non_repeating_elements2 = set2 - common_elements

    # Combine the non-repeating elements from both sets
    result = sorted(list(non_repeating_elements1.union(non_repeating_elements2)))

    # Calculate the total number of non-repeating elements
    total_non_repeating = len(result)

    return result, total_non_repeating

# Read the input
arr1_size, arr2_size = map(int, input().split())
```

```
arr1_size, arr2_size = map(int, input().split())
arr1 = list(map(int, input().split()))
arr2 = list(map(int, input().split()))

# Find the non-repeating elements
result, total_non_repeating = find_non_repeating_elements(arr1, arr2)

# Print the output
if total_non_repeating == 0:
    print("NO SUCH ELEMENTS")
else:
    print(*result)
    print(total_non_repeating)

```

	Input	Expected	Got	
✓	5 4 1 2 8 6 5 2 6 8 10	1 5 10 3	1 5 10 3	✓
✓	3 3 10 10 10 10 11 12	11 12 2	11 12 2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

The **DNA sequence** is composed of a series of nucleotides abbreviated as `'A'`, `'C'`, `'G'`, and `'T'`.

- For example, `"ACGAATTCCG"` is a **DNA sequence**.

When studying **DNA**, it is useful to identify repeated sequences within the DNA.

Given a string `s` that represents a **DNA sequence**, return all the `10`**-letter-long** sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in **any order**.

Example 1:

```
Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]
```

Example 2:

```
Input: s = "AAAAAAAAAAAAA"
Output: ["AAAAAAAAAA"]
```

For example:

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA

Answer: (penalty regime: 0 %)

```
s = input()
def findRepeatedSequences(s):
    seen = set()
    repeated = set()
    result = []

    for i in range(len(s) - 9):
        sequence = s[i:i+10]
        if sequence in seen:
            repeated.add(sequence)
        else:
            seen.add(sequence)

    for seq in repeated:
        result.append(seq)

    return result
r = findRepeatedSequences(s)

for i in r:
    print(i)
```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA	AAAAACCCCC CCCCCAAAAA	✓

	Input	Expected	Got	
✓	AAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Coders here is a simple task for you, Given string str. Your task is to check whether it is a binary string or not by using python set.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No

Answer: (penalty regime: 0 %)

```
str = input()
binary_set = set("01")
if all(char in binary_set for char in str):
    print("Yes")
else:
    print("No")

```

	Input	Expected	Got	
✓	01010101010	Yes	Yes	✓
✓	REC123	No	No	✓
✓	010101 10101	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.There is only **one repeated number** in `nums`, return *this repeated number*. Solve the problem using set.

Example 1:

```
Input: nums = [1,3,4,2,2]
```

```
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
```

```
Output: 3
```

For example:

Input	Result
1 3 4 4 2	4

Answer: (penalty regime: 0 %)

```
def findDuplicate():
    nums = input().split()
    nums = [int(num) for num in nums]

    num_set = set()

    for num in nums:
        if num in num_set:
            return num
        num_set.add(num)

# Get user input
duplicate_number = findDuplicate()
print(duplicate_number)
```

	Input	Expected	Got	
✓	1 3 4 4 2	4	4	✓
✓	1 2 2 3 4 5 6 7	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week7_MCQ

Jump to...

Dictionary ►