Dashboard / My courses / PSPP/PUP / Experiments based on Tuples, Sets and its operations / Week7_Coding

Started on	Wednesday, 5 June 2024, 1:06 PM
State	Finished
Completed on	Wednesday, 5 June 2024, 9:34 PM
Time taken	8 hours 27 mins
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 %)

```python
def count_distinct_pairs(t, K):
    freq_map = {}
    distinct_pairs = set()

    for num in t:
        complement = K - num
        if complement in freq_map:
            pair = tuple(sorted((num, complement)))
            distinct_pairs.add(pair)

        if num in freq_map:
            freq_map[num] += 1
        else:
            freq_map[num] = 1

    return len(distinct_pairs)

# Example usage
t = tuple(map(int,input().split(',')))
K = int(input())

print(count_distinct_pairs(t, K))

```

	Input	Expected	Got	
✓	5,6,5,7,7,8 13	2	2	✓
✓	1,2,1,2,5 3	1	1	✓
✓	1,2 0	0	0	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

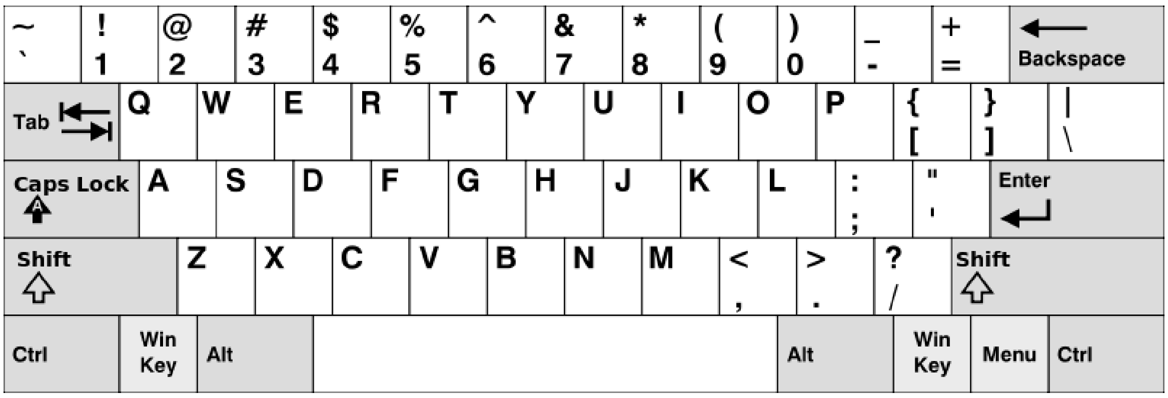
Correct

Mark 1.00 out of 1.00

Given an array of strings `words`, return *the words that can be typed using letters of the alphabet on only one row of American keyboard like the image below.*

In the **American keyboard**:

- the first row consists of the characters `"qwertyuiop"`,
- the second row consists of the characters `"asdfghjkl"`, and
- the third row consists of the characters `"zxcvbnm"`.


~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = Backspace
Tab Q W E R T Y U I O P { [}] | \
Caps Lock A S D F G H J K L : ; " ' Enter
Shift Z X C V B N M < , > . ? / Shift
Ctrl Win Key Alt Alt Win Key Menu Ctrl

Example 1:

```
Input: words = ["Hello","Alaska","Dad","Peace"]
Output: ["Alaska","Dad"]
```

Example 2:

```
Input: words = ["omk"]
Output: []
```

Example 3:

```
Input: words = ["adsdf","sfd"]
Output: ["adsdf","sfd"]
```

For example:

Input	Result
4 Hello Alaska Dad Peace	Alaska Dad
2 adsfd afd	adsfd afd

Answer: (penalty regime: 0 %)

```
def findwords(words):
    row1 = set('qwertyuiop')
    row2 = set('asdfghjkl')
    row3 = set('zxcvbnm')
    result = []
    for word in words:
       w = set(word.lower())
       if w.issubset(row1) or w.issubset(row2) or w.issubset(row3):
           result.append(word)
    if len(result) ==0:
        print("No words")
    else:
        for i in result:
            print(i)
a=int(input())
arr = [input() for i in range(a)]
findwords(arr)
```

	Input	Expected	Got	
✓	4 Hello Alaska Dad Peace	Alaska Dad	Alaska Dad	✓
✓	1 omk	No words	No words	✓
✓	2 adsfd afd	adsfd afd	adsfd afd	✓

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()
substring_counts={}
for i in range(len(s)-9):
    substring=s[i:i+10]
    substring_counts[substring]=substring_counts.get(substring,0)+1
repeated_substrings=[substring for substring, count in substring_counts.items() if count>1]
for substring in repeated_substrings:
    print(substring)
```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA	AAAAACCCCC CCCCCAAAAA	✓
✓	AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

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

5 5
1 2 3 4 5
1 2 3 4 5
	NO SUCH ELEMENTS

Answer: (penalty regime: 0 %)

```python
def find_non_repeating_elements():
    n,m=map(int, input().split())
    arr1=list(map(int, input().split()))
    arr2=list(map(int, input().split()))
    set1=set(arr1)
    set2=set(arr2)
    non_repeating_elements = set1.symmetric_difference(set2)
    if len(non_repeating_elements) == 0:
        print("NO SUCH ELEMENTS")
    else:
        print(' '.join(map(str, non_repeating_elements)))
        print(len(non_repeating_elements))
find_non_repeating_elements()
```

	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	✓
✓	5 5 1 2 3 4 5 1 2 3 4 5	NO SUCH ELEMENTS	NO SUCH ELEMENTS	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

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 %)

```
a=input()
try:
    int(a)
    print("Yes")
except:
    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.

◄ Week7_MCQ

Jump to...

Dictionary ►