

[Dashboard](#) / [My courses](#) / [PSP/PUP](#) / [Experiments based on Tuples, Sets and its operations](#) / [Week7 Coding](#)

| | |
|---------------------|------------------------------|
| Started on | Friday, 7 June 2024, 9:37 PM |
| State | Finished |
| Completed on | Friday, 7 June 2024, 9:43 PM |
| Time taken | 5 mins 47 secs |
| Marks | 5.00/5.00 |
| Grade | 100.00 out of 100.00 |

Question 1

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 %)

```
1 def find_non_repeating_elements():
2     n,m=map(int, input().split())
3     arr1=list(map(int, input().split()))
4     arr2=list(map(int, input().split()))
5     set1=set(arr1)
6     set2=set(arr2)
7     non_repeating_elements = set1.symmetric_difference(set2)
8     if len(non_repeating_elements) == 0:
9         print("NO SUCH ELEMENTS")
10    else:
11        print(' '.join(map(str, non_repeating_elements)))
12        print(len(non_repeating_elements))
13    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 **2**

Correct

Mark 1.00 out of 1.00

There is a malfunctioning keyboard where some letter keys do not work. All other keys on the keyboard work properly.

Given a string text of words separated by a single space (no leading or trailing spaces) and a string brokenLetters of all distinct letter keys that are broken, return the number of words in text you can fully type using this keyboard.

Example 1:

Input: text = "hello world", brokenLetters = "ad"

Output:

1

Explanation: We cannot type "world" because the 'd' key is broken.

For example:

| Input | Result |
|--|--------|
| hello world ad | 1 |
| Faculty Upskilling in Python Programming ak | 2 |

Answer: (penalty regime: 0 %)

```

1 a=list(input().split())
2 b=list(input())
3 c=0
4 for i in a:
5     d=0
6     for j in b:
7         if j in i.lower():
8             d+=1
9     if d == 0:
10        c+=1
11 print(c)

```

| | Input | Expected | Got | |
|---|--|----------|-----|---|
| ✓ | hello world ad | 1 | 1 | ✓ |
| ✓ | Welcome to REC e | 1 | 1 | ✓ |
| ✓ | Faculty Upskilling in Python Programming ak | 2 | 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, "ACGAATTCG" 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 = "AAAAACCCCCAAAAACCCCCAAAAAGGGTTT"`Output: `["AAAAACCCC", "CCCCAAAAA"]`**Example 2:**Input: `s = "AAAAAAAAAAAA"`Output: `["AAAAAAAAA"]`

For example:

| Input | Result |
|---------------------------------|------------------------|
| AAAAACCCCCAAAAACCCCCAAAAAGGGTTT | AAAAACCCC CCCCAAAAA |

Answer: (penalty regime: 0 %)

```

1 s=input()
2 substring_counts={}
3 for i in range(len(s)-9):
4     substring=s[i:i+10]
5     substring_counts[substring]=substring_counts.get(substring,0)+1
6 repeated_substrings=[substring for substring,count in substring_counts.items() if count>1]
7 for substring in repeated_substrings:
8     print(substring)

```

| | Input | Expected | Got | |
|---|---------------------------------|------------------------|------------------------|---|
| ✓ | AAAAACCCCCAAAAACCCCCAAAAAGGGTTT | AAAAACCCC CCCCAAAAA | AAAAACCCC CCCCAAAAA | ✓ |
| ✓ | AAAAAAAAAAAA | AAAAAAAAA | AAAAAAAAA | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question 4

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 %)

```

1  #Input tuple and K value
2  t=tuple(map(int,input().split(',')))
3  k=int(input())
4  s=set(t)
5  count=0
6  for x in s:
7      if k-x in s:
8          count+=1
9  result=count//2
10 print(result)
11

```

| | 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 5

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 afd | adsfd afd |

Answer: (penalty regime: 0 %)

```

1 def findwords(words):
2     row1 = set('qwertyuiop')
3     row2 = set('asdfghjkl')
4     row3 = set('zxcvbnm')
5     result = []
6     for word in words:
7         w = set(word.lower())
8         if w.issubset(row1) or w.issubset(row2) or w.issubset(row3):
9             result.append(word)
10    if len(result) == 0:
11        print("No words")
12    else:

```



```
13 |         for i in result:
14 |             print(i)
15 | a = int(input())
16 | arr = [input() for i in range(a)]
17 | 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.

◀ Week7_MCQ

Jump to...

Dictionary ▶