Started on	Thursday, 6 June 2024, 8:26 PM
State	Finished
Completed on	Friday, 7 June 2024, 6:16 PM
Time taken	21 hours 50 mins
Marks	4.00/5.00
Grade	**80.00** out of 100.00

Question **1**

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 %)

```
def is_binary_string(s):
    binary_set = {'0', '1'}
    return set(s) <= binary_set

def main():
    string = input().strip()
    if is_binary_string(string):
        print("Yes")
    else:
        print("No")

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

	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 **2**

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 %)

```
# Take use input for the array
nums = list(map(int, input().split()))

#set to store seen numbers
num_set = set()

# Iterate through the array and find the duplicate element
for num in nums:
    if num in num_set:
        print(num)
        break
    else:
        num_set.add(num)
```

	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.

Question **3**

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 %)

```
def find_non_repeating(arr1, arr2):
    unique_elements = set(arr1) ^ set(arr2)
    if unique_elements:
        return sorted(unique_elements), len(unique_elements)
    else:
        return "NO SUCH ELEMENTS"

def main():
    size1, size2 = map(int, input().split())
    arr1 = list(map(int, input().split()))
    arr2 = list(map(int, input().split()))

    result = find_non_repeating(arr1, arr2)
    if isinstance(result, tuple):
        print(*result[0])
        print(result[1])
    else:
        print(result)
```

```
        print(result)

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

	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 **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 %)

```
def count_distinct_pairs(t, K):
    seen = set()
    pairs = set()
    for x in t:
        complement = K - x
        if complement in seen:
            pair = (min(x, complement), max(x, complement))
            pairs.add(pair)
        seen.add(x)
    return len(pairs)

def main():
    t = tuple(map(int, input().split(',')))
    K = int(input())
    result = count_distinct_pairs(t, K)
    print(result)

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

	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**

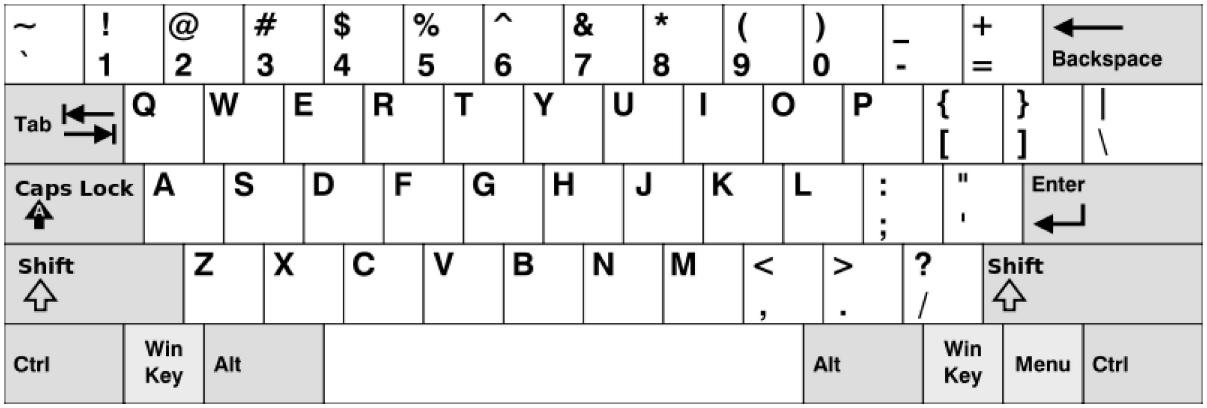
Incorrect

Mark 0.00 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"`.



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 find_words(words):
    rows = [
        set("qwertyuiop"),
        set("asdfghjkl"),
        set("zxcvbnm")
```

```
    ]
    result = []
    for word in words:
        word_lower = word.lower()
        in_one_row = True
        for row in rows:
            if all(char in row for char in word_lower):
                result.append(word)
                break
    return result

# Example usage:
n = int(input().strip())
words = [input().strip() for _ in range(n)]
print(find_words(words))
```

	Input	Expected	Got	
✗	4 Hello Alaska Dad Peace	Alaska Dad	['Alaska', 'Dad']	✗
✗	1 omk	No words	[]	✗
✗	2 adsfd afd	adsfd afd	['adsfd', 'afd']	✗

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

◄ Week7_MCQ

Jump to...

Dictionary ►