08 – Tuple/Set

Binary String

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:

```
str1=set(input())
if not(str1-{'0','1'}):
    print("Yes")
else:
    print("No")
```

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

<u>Check Pair</u>

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:

```python
def find_pairs_with_sum(numbers, target_sum):
    numbers_list = list(numbers)
    pairs = set()
    visited = set()
    for number in numbers_list:
        complement = target_sum - number
        if complement in visited:
            pair = tuple(sorted((number, complement)))
            pairs.add(pair)
        visited.add(number)
    return pairs
numbers_input = input("")
target_sum = int(input(""))
numbers = tuple(map(int, numbers_input.split(',')))
pairs = find_pairs_with_sum(numbers, target_sum)
print(f"{len(pairs)}")
```

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

DNA Sequence

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 = "AAAAACCCCCAAAAACCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]
Example 2:

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

For example:

Input	Result
AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA

ANSWER:

```
a=input()
b=[]
for i in range(0,len(a),10):
    b.append(a[i:i+10])
print(b[0])
for i in range(len(b)-1):
    if(b[i]==b[i+1]):
        print(b[i+1][::-1])
```

Input	Expected	Got	
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA	AAAAACCCCC CCCCCAAAAA	
AAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	

Print repeated no

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:

```python
a=list(input().split(" "))
a=[int(x) for x in a]
for i in a:
    if a.count(i)>1:
        print(i)
        break
```

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

Remove repeated

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:

```python
def find_non_repeating_elements(arr1, arr2):
operations
    set1 = set(arr1)
    set2 = set(arr2)
  non_repeating_elements = (set1 - set2).union(set2 - set1)
  non_repeating_elements = sorted(list(non_repeating_elements))
   if non_repeating_elements:
      print(" ".join(map(str, non_repeating_elements)))
      print(len(non_repeating_elements))
   else:
      print("NO SUCH ELEMENTS")
sizes = input().split()
size1 = int(sizes[0])
size2 = int(sizes[1])
array1 = list(map(int, input().split()))
array2 = list(map(int, input().split()))
find_non_repeating_elements(array1, array2)
```

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	

Malfunctioning Keyboard

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

ANSWER:

```python
def count_typeable_words(text, brokenLetters):
    broken_set = set(brokenLetters.lower())
    words = text.split()
    count = 0
    for word in words:
        if not any(char.lower() in broken_set for char in word):
            count += 1
            return count
text = input()
brokenLetters = input()
result = count_typeable_words(text, brokenLetters)
print(result)
```

Input	Expected	Got
hello world ad	1	1
Welcome to REC e	1	1
Faculty Upskilling in Python Programming ak	2	2

American keyboard

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

ANSWER:

```python
def find_words(words):
    row1 = set("qwertyuiop")
    row2 = set("asdfghjkl")
    row3 = set("zxcvbnm")
    result = []
    for word in words:
        lower_word = set(word.lower())
        if lower_word.issubset(row1) or lower_word.issubset(row2) or lower_word.issubset(row3):
            result.append(word)
    return result
n = int(input())
input_words = [input().strip() for _ in range(n)]
result = find_words(input_words)
if result:
    for word in result:
        print(word)
else:
    print("No words")
```

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	