**08 – Tuple/Set**

# Ex. No. : 8.1 Date:

# Register No.: Name:

# 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 |

# PROGRAM:

# a = input()

# try:

# c = int(a)

# print("Yes")

# except:

# print("No")

# 

# Ex. No. : 8.2 Date:

# Register No.: Name:

# Check Pair

# 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 |
| --- | --- |
| 1,2,1,2,53 | 1 |
| 1,20 | 0 |

# PROGRAM:

# t = input().split(',')

# k = int(input())

# s = []

# s1=[]

# for i in range(len(t)-1):

# for j in range(i+1,len(t)):

# if (int(t[i])+int(t[j]))==k and (t[i],t[j]) not in s1 and (t[i],t[j]) not in s1:

# s1.append((t[i],t[j]))

# s1.append((t[j],t[i]))

# s.append((t[i],t[j]))

# 

# Ex. No. : 8.3 Date:

# Register No.: Name:

# 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 = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

# Output: ["AAAAACCCCC","CCCCCAAAAA"]

# Example 2:

# Input: s = "AAAAAAAAAAAAA"

# Output: ["AAAAAAAAAA"]

# 

# 

# For example:

| Input | Result |
| --- | --- |
| AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT | AAAAACCCCCCCCCCAAAAA |

# PROGRAM:

# def findRepeatedSequences(s):

# sequences = {}

# result = []

# for i in range(len(s) - 9):

# seq = s[i:i+10]

# sequences[seq] = sequences.get(seq, 0) + 1

# if sequences[seq] == 2:

# result.append(seq)

# return result

# # Example usage

# s1 = input()

# for i in findRepeatedSequences(s1):

# print(i)

# 

# 

# Ex. No. : 8.4 Date:

# Register No.: Name:

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 |

# PROGRAM:

# 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)

# 

# Ex. No. : 8.5 Date:

# Register No.: Name:

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 |

PROGRAM:

def can\_type(text, brokenLetters):

words = text.split()

valid\_word\_count = 0

for word in words:

valid = True

for letter in word:

letter=letter.lower()

if letter in brokenLetters:

valid = False

break

if valid:

valid\_word\_count += 1

return valid\_word\_count

text = input()

brokenLetters = input()

print(can\_type(text, brokenLetters))

