1) 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

PROGRAM:

def is\_binary\_string(input\_str):

return set(input\_str) <= {'0', '1'}

input\_str1 = input()

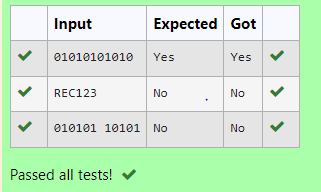
if is\_binary\_string(input\_str1):

print("Yes")

else:

print("No")

OUTPUT:



2) 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"]

PROGRAM:

def findRepeatedSequences(s):

seen = set()

repeated = []

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

sequence = s[i:i+10]

if sequence in seen and sequence not in repeated:

repeated.append(sequence)

else:

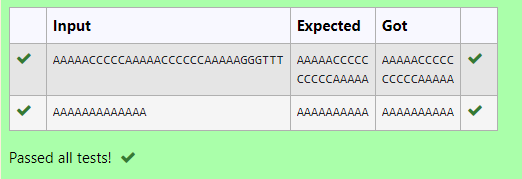
seen.add(sequence)

return "\n".join(repeated)

s1 = input()

print(findRepeatedSequences(s1))

OUTPUT:



3) 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.

PROGRAM:

count=0

a=map(int,input().split(","))

k=int(input())

a=list(set(a))

for i in range(len(a)):

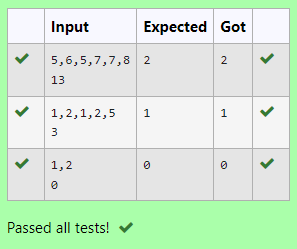
for j in range(i+1,len(a)):

if a[i]+a[j]==k:

count+=1

print(count)

OUTPUT:



4) 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](about:blank) Input:

5 4

1 2 8 6 5

2 6 8 10

[Sample](about:blank) Output:

1 5 10

3

[Sample](about:blank)  Input:

5 5

1 2 3 4 5

1 2 3 4 5

[Sample](about:blank) Output:

NO SUCH ELEMENTS

PROGRAM:

def find\_non\_repeating\_elements(arr1, arr2):

unique\_elements = set(arr1) ^ set(arr2)

non\_repeating\_elements = sorted(list(unique\_elements))

return non\_repeating\_elements, len(non\_repeating\_elements)

# Input

size1, size2 = map(int, input().split())

arr1 = list(map(int, input().split()))

arr2 = list(map(int, input().split()))

# Output

non\_repeating, count = find\_non\_repeating\_elements(arr1, arr2)

if count > 0:

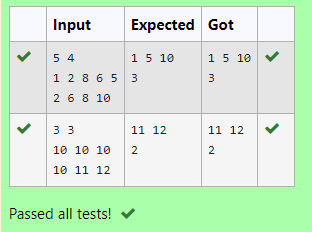
print(\*non\_repeating)

print(count)

else:

print("NO SUCH ELEMENTS")

OUTPUT:



5) 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.

PROGRAM:

def count\_words\_typed(text, brokenLetters):

count = 0

for word in text.split():

if all(letter not in brokenLetters for letter in word):

count += 1

return count

# Test the function

text = input().lower()

brokenLetters = input()

print(count\_words\_typed(text, brokenLetters)) # Output: 1

OUTPUT:

