WEEK 7:

1.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

**Program**:

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

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

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

set1 = set(arr1)

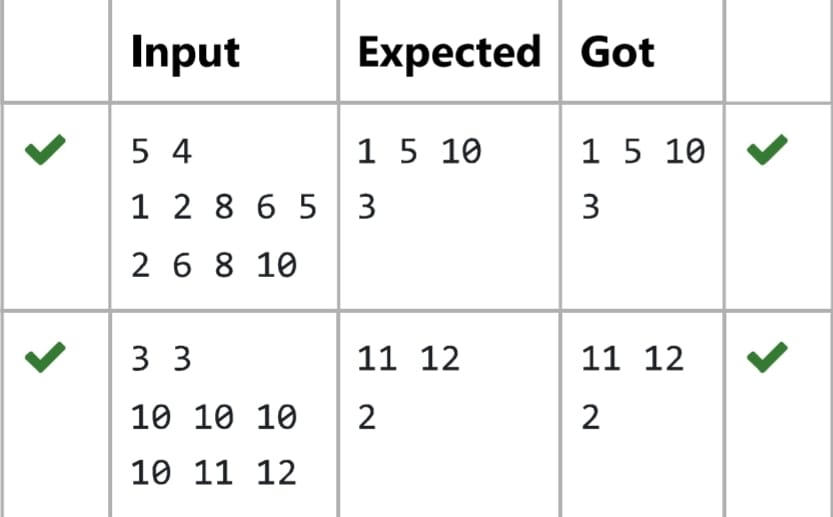
set2 = set(arr2)

non\_repeating = set1.symmetric\_difference(set2)

print(\*non\_repeating)

print(len(non\_repeating))

**Output**:



2.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

**Program**:

text = input()

brokenLetters = input()

words = text.split()

count = 0

for word in words:

word = word.lower()

if any(letter in brokenLetters for letter in word):

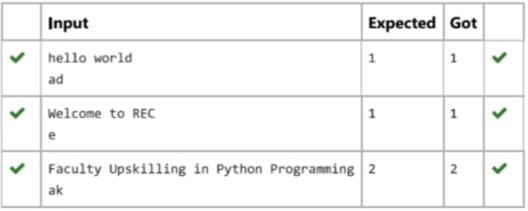
continue

else:

count += 1

print(count)

**Output**:



3.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.

**Program**:

s = input()

substrings = {}

repeated\_substrings = []

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

substring = s[i:i + 10]

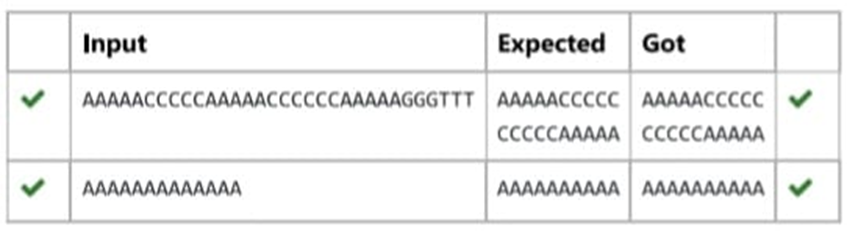
substrings[substring] = substrings.get(substring, 0) + 1

if substrings[substring] == 2:

repeated\_substrings.append(substring)

print(\*repeated\_substrings, sep="\n")

**Output**:



4. 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"]

rows = ['qwertyuiop', 'asdfghjkl', 'zxcvbnm']

**Program**:

n = int(input())

words = [input() for \_ in range(n)]

result = []

for word in words:

lowercase\_word = word.lower()

same\_row = False

for row in rows:

if all(char in row for char in lowercase\_word):

same\_row = True

break

if same\_row:

result.append(word)

if result:

for word in result:

print(word)

else:

print("No words")

**Output**:



5. 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**:

string=input()

s=set(string)

if s=={'0', '1') or s=={'0'} or s=={'1'}:

print("Yes")

else:

print("No")

**Output**:

