**Week 7 - Experiments based on Tuples, Sets and its operations.**

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

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

PROGRAM:

def find\_non\_repeating\_elements():

n, m = map(int, input().split())

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

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

set1 = set(arr1)

set2 = set(arr2)

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

if len(non\_repeating\_elements) == 0:

print("NO SUCH ELEMENTS")

else:

print(' '.join(map(str, non\_repeating\_elements)))

print(len(non\_repeating\_elements))

find\_non\_repeating\_elements()

OUTPUT:



2. 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

PROGRAM:

n =input().split(" ")

n = list(n)

for i in range(len(n)):

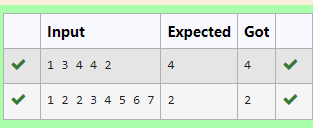
for j in range(i+1,len(n)):

if n[i] == n[j]:

print(n[i])

exit(0)

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.

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAAA"

Output: ["AAAAAAAAAA"]

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)

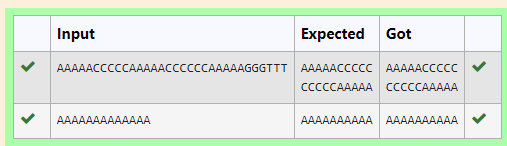
return result

s1 = input()

for i in findRepeatedSequences(s1):

print(i)

OUTPUT:



4. 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 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

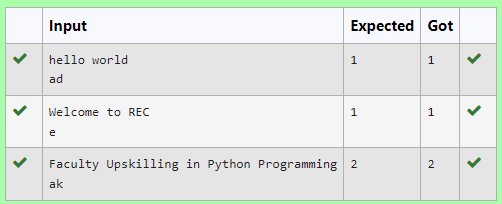
return valid\_word\_count

text = input()

brokenLetters = input()

print(can\_type(text, brokenLetters))

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

**For example:**

| **Input** | **Result** |
| --- | --- |
| 01010101010 | Yes |
| 010101 10101 | No |

PROGRAM:

a = input()

try:

c = int(a)

print("Yes")

except:

print("No")

OUTPUT:

