**WEEK-7 CODING**

1.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

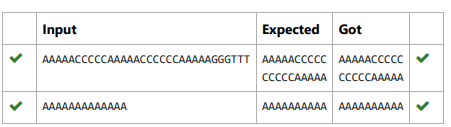
# Example usage

s1 = input()

for i in findRepeatedSequences(s1):

print(i)

**OUTPUT:**



2. 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

**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\_elements = set1.symmetric\_difference(set2)

if non\_repeating\_elements:

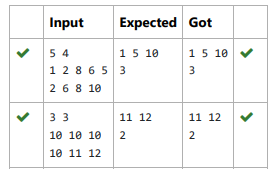
print(\*sorted(non\_repeating\_elements))

print(len(non\_repeating\_elements))

else:

print("NO SUCH ELEMENTS")

**OUTPUT:**

****

3. 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"]

**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)

**OUTPUT:**

****

4. 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:**

a = input()

try:

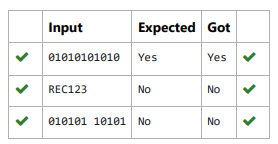
c = int(a)

print("Yes")

except:

print("No")

**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 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

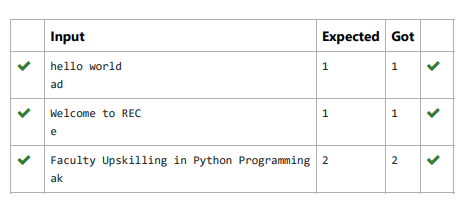
return valid\_word\_count

text = input()

brokenLetters = input()

print(can\_type(text, brokenLetters))

**OUTPUT:**

****