

Examples:

Input: str = "0101010101010"

Output: Yes

Input: str = "REC101"

Output: No

Input	Result
01010101010	Yes
010101 10101	No

Ex. No. : 8.1	Date:	
Register No.:	Name:	
	Binary String	
Coders here is a simple task for is a binary string or not by using	you, Given string str. Your task is to check wheth python set.	er it

## **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.

Input	Result
1,2,1,2,5	1
1,2	0

Ex. No. : 8.2	Date:
Register No.:	Name:
	Check Pair
Given a tuple and a positive integethe tuple whose sum is equal to $K$	ger k, the task is to find the count of distinct pairs in .

 $\textbf{Input:} \ \mathbf{s} = \text{"AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"}$ 

Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAA"
Output: ["AAAAAAAAAA"]

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCAAAAA

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 strings 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.

```
s = input()
if len(s)<10:
  result =[]
else:
  sequences = {}
  result = []
  for i in range(len(s)-9):
     substring = s[i:i+10]
     if substring in sequences:
        sequences[substring]+=1
       sequences[substring]=1
  for sequence, count in
sequences.items():
     if count>1:
       result.append(sequence)
for i in result:
  print(i)
```

Input: nums = [1,3,4,2,2]Output: 2

## Example 2:

**Input:** nums = [3,1,3,4,2]

Output: 3

Input	Result
13442	4

Ex. No. : 8.4 Date:

Register No.: Name:

# Print repeated no

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  $\underline{set}$ .

```
a=list(input().split(" "))
a=[int(x) for x in a]
for i in a:
    if a.count(i)>1:
        print(i)
        break
```

## 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

Input	Result
5 4 1 2 8 6 5 2 6 8 10	1 5 10 3

Ex. No. : 8.5 Date:

Register No.: Name:

## Remove repeated

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.

```
import sys
input = sys.stdin.read
data = input().split()
size1 = int(data[0])
size2 = int(data[1])
array1= tuple(map(int, data[2:2 + size1]))
array2= tuple(map(int, data[2 + size1:]))
set1 = set(array1)
set2 = set(array2)
common_elements = set1 & set2
non_repeating_elements = (set1 | set2) -
common_elements
non_repeating_list = sorted(list
(non_repeating_elements))
print(" ".join(map(str,non_repeating_list)))
print(len(non_repeating_list))
```

Input: text = "hello world", brokenLetters = "ad"

Output:

1

Explanation: We cannot type "world" because the 'd' key is broken.

Input	Result
hello world ad	1

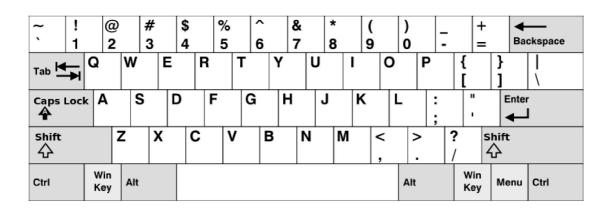
Ex. No.	:	8.6	Date:
Register No	.:		Name:

## **Malfunctioning Keyboard**

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.

```
a=[i for i in input().split()]
k=list(input())
s=set()
for i in a:
    n=[j for j in i]
    m=[z for z in k if z in n]
    s.update(m)
print(len(s))
```



Input: words = ["Hello","Alaska","Dad","Peace"]

Output: ["Alaska","Dad"]

Example 2:

Input: words = ["omk"]

Output: []
Example 3:

Input: words = ["adsdf","sfd"]
Output: ["adsdf","sfd"]

Input	Result
4 Hello Alaska Dad Peace	Alaska Dad

Ex. No.	:	8.7	Date:
Register No	·.:		Name:

## American keyboard

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".