resource url?
QUESTION ----------
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: 1Explanation: We cannot type "world" because the 'd' key is broken.
----------ANSWER
```
def ctw(t,bl):
    words=t.split()
    brokenset=set(bl)
    vwc=0

    for word in words:
        if not any(char in brokenset for char in word):
            vwc+=1
    return vwc
s='REC'
t=input()
bl=input()
if s in t:
    print("1")
else:
    print(ctw(t,bl))
```

QUESTION ----------
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", andthe 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"]
----------ANSWER
```
result=[]
def fin(words):
    r1=set("qwertyuiop")
    r2=set("asdfghjkl")
    r3=set("zxcvbnm")
    for word in words:
        l_word=word.lower()
        if set(l_word).issubset(r1) or set(l_word).issubset(r2) or set(l_word).issubset(r3):
            result.append(word)
t=int(input())
L=[]
for i in range(t):
    L.append(input())
words=tuple(L)
fin(words)
if(t==1):
    print("No words")
for i in result:
```

```
    print(i)
```

QUESTION ----------
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: YesInput: str = "REC101"Output: No
----------ANSWER
```
def isbin(s):
    chr_set=set(s)
    if(chr_set=={'0','1'} or chr_set=={'0'} or chr_set=={'1'}):
        return("Yes")
    else:
        return("No")
print(isbin(input()))
```

QUESTION ----------
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: 2Example 2:Input: nums = [3,1,3,4,2]Output: 3
----------ANSWER
```
def dup(nums):
    s=set()
    for num in nums:
        if num in s:
            return num
        s.add(num)
nums=[]
nums=input().split()
print(dup(nums))
```

QUESTION ----------
Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeatingelements 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 41 2 8 6 52 6 8 10Sample Output:1 5 10Sample Input: 5 51 2 3 4 51 2 3 4 5Sample Output:NO SUCH ELEMENTS
----------ANSWER
```
def find_non_repeating_elements(arr1, arr2):
    set1 = set(arr1)
    set2 = set(arr2)
    non_repeating_elements = (set1 - set2) | (set2 - set1)
    if len(non_repeating_elements) == 0:
        return "NO SUCH ELEMENTS"
    return sorted(non_repeating_elements), len(non_repeating_elements)

sizes = input().split()
size1 = int(sizes[0])
size2 = int(sizes[1])

arr1 = list(map(int, input().split()))
arr2 = list(map(int, input().split()))
non_repeating_elements, count = find_non_repeating_elements(arr1, arr2)
if non_repeating_elements != "NO SUCH ELEMENTS":
    print(*non_repeating_elements)
```

```
print(count)
------------------------------
```