08 – Tuple/Set

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No

Ex. No. : 8.1 **Date:**

Register No.: **Name:**

Binary String

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.

```
n=input()
for i in n:
    if i not in ['1','0']:
        print("No")
        break
else:
    print("Yes")
```

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.

For example:

Input	Result
1,2,1,2,5 3	1
1,2 0	0

Ex. No. : 8.2 **Date:**

Register No.: **Name:**

Check Pair

Given a tuple and a positive integer k, the task is to find the count of distinct pairs in the tuple whose sum is equal to **K**.

```
a=tuple(input())
k=int(input())
b=list(set([int(i) for i in a if i!=','']))
c=0
for i in range(0,len(b)):
    for j in range(i+1,len(b)):
        #print(i,j)
        if b[i]+b[j]==k:
            c+=1
print(c)
```

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

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

For example:

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA

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 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**.

```
def findRepeatedDnaSequences (s):
    if len(s)<10:
        return []
    sequences = {}
    result =[]
    for i in range (len(s)-9):
        substring=s[i:i+10]
        if substring in sequences:
            sequences[substring] +=1
        else:
            sequences [substring]=1

    for sequence, count in sequences.items():
        if count > 1:
            result.append (sequence)

    for i in result:
        print(i)
```

```
s1=input()
findRepeatedDnaSequences (s1)
```

Example 1:

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

Example 2:

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

For example:

Input	Result
1 3 4 4 2	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 set.

```
nums=input().split()
for i in nums:
    if nums.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

For example:

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.

```
l=input()
l=l.split(" ")
l1=int(l[0])
l2=int(l[1])
c=[]
a=input()
b=input()
a=a.split(" ")
b=b.split(" ")
for i in range(l1):
    if(a[i] not in b):
        c.append(a[i])
for i in range(l2):
    if(b[i] not in a):
        c.append(b[i])
```

```
for i in range(len(c)):
    print(c[i],end=" ")
print()
print(len(c))
```

Example 1:

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

Output:

1

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

For example:

Input	Result
hello world ad	1

Ex. No. : 8.6 **Date:**

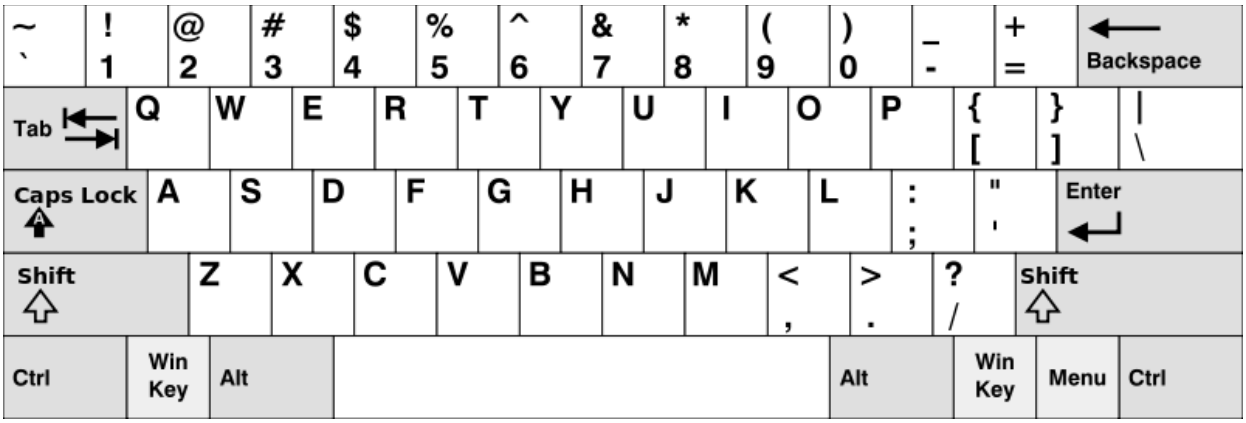
Register No.: **Name:**

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.

```
s1=list(input())
s2=list(input())
b=[]
count=0
for i in range(len(s1)):
    for j in range(len(s2)):
        if s2[j] in s1[i]:
            if s2[j] not in b:
                b.append(s2[j])
for i in range(len(b)):
    count+=1
print(count)
```



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"]

For example:

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".

```
a=int(input())

c=[]

l1="qwertyuiop"

l2="asdfghjkl"

l3="zxcvbnm"

c1,c2,c3=0,0,0

r=0

for i in range(a):

    b=input()

    c.append(b)

for i in range(len(c)):



    for j in range(len(c[i])):

        if((c[i][j] in l1 ) ):

            c1+=1
```

```
            if(c1==len(c[i])):
                print(c[i])
                c1=0
                r+=1
        elif ( c[i][j] in l2 ):
            c2+=1
            if(c2==len(c[i])):
                print(c[i])
  c2=0
                r+=1
        elif ( c[i][j] in l3):
            c3+=1
            if(c3==len(c[i])):
                print(c[i])
                c3=0
                r+=1
if r==0:
    print("No words")
```