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.: 230701357 **Name: SWETHA.J**

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.

Program:

```
n=input()

a=set(n)

if a=={'0','1'}:

    print("Yes")

else:

    print("No")
```

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.: 230701357 **Name: SWETHA.J**

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**.

Program:

```
t=str(input())
tu=t.split(',')
k=int(input())
numbers=[]
for i in tu:
    if i.isdigit():
        numbers.append(int(i))
d=0
s=set()
pairs=set()
for num in numbers:
    c=k-num
    if c in s:
        pairs.add((min(num,c),max(num,c)))
    s.add(num)
d=len(pairs)
print(d)
```

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.: 230701357 **Name: SWETHA.J**

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**.

Program:

```
s = input()
sequences = set()
repeated_sequences = set()

for i in range(len(s) - 9):
    sequence = s[i:i+10]
    if sequence in sequences:
        repeated_sequences.add(sequence)
    else:
        sequences.add(sequence)

for sequence in repeated_sequences:
    print(sequence)
```

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.: 230701357 **Name: SWETHA.J**

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.

Program:

```
nums = [int(x) for x in input().split()]

seen = set()

for num in nums:

    if num in seen:

        print(num)

        break

    seen.add(num)
```

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.: 230701357 **Name: SWETHA.J**

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.

Program:

```
s = input()

a, b = s.split()

c = int(a)d = int(b)

al = input()

a2 = input()

bl = al.split()

b2 = a2.split()

c1 = list(map(int, bl))

c2 = list(map(int, b2))

set1 = set(c1)

set2 = set(c2)

n = set1.symmetric_difference(set2)

if n:

    print(*n)
```

```
    print(len(n))
else:
    print("NO SUCH ELEMENTS")
```

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.: 230701357 **Name: SWETHA.J**

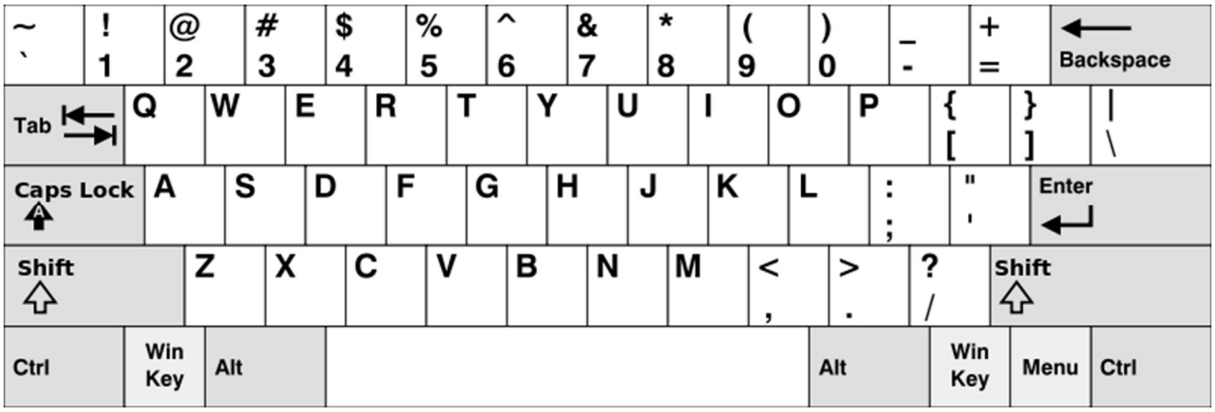
Malfunctioning Keyboard

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.

Program:

```
n=input()
b=n.lower()
a=input()
word=b.split()
count=0
for i in word:
    if any(letter in a for letter in i):
        continue
    else:
        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.: 230701357 **Name: SWETHA.J**

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".

Program:

```
key=['qwertyuiop', 'asdfghjkl','zxcvbnm']

n=int(input())

words=[]

results=[]

for i in range(n):

    a=input()

    words.append(a)

for word in words:

    lower= word.lower()

    found=False

    for row in key:

        if all(letter in row for letter in lower):

            found=True

            break
```

```
    if found:
        results.append(word)
if(len(results)==0):
    print("No words")
else:
    for i in results:
        print(i)
```