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.

```
bi = ["0","1"]
s = input()
flag = 0
for i in s:
    if i not in bi:
        flag = 1
        break
if flag==1:
    print("No")
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**.

```
def find_pairs_with_sum(numbers, target_sum):
    numbers_list = list(numbers)
    pairs = set()
    visited = set()
    for number in numbers_list:
        complement = target_sum - number
        if complement in visited:
            pair = tuple(sorted((number, complement)))
            pairs.add(pair)
        visited.add(number)
    return pairs
numbers_input = input("")
target_sum = int(input(""))
numbers = tuple(map(int, numbers_input.split(',')))
pairs = find_pairs_with_sum(numbers, target_sum)
print(f"{len(pairs)}")
```

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**.

```
x=input()
s=set()
r=set()
for i in range(len(x)-9):
    ss=x[i:i+10]
    if ss in s:
        r.add(ss)
    else:
        s.add(ss)
list(r)
for j in r:
    print(j)
```

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.

```
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

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.

```
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))
```

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:**

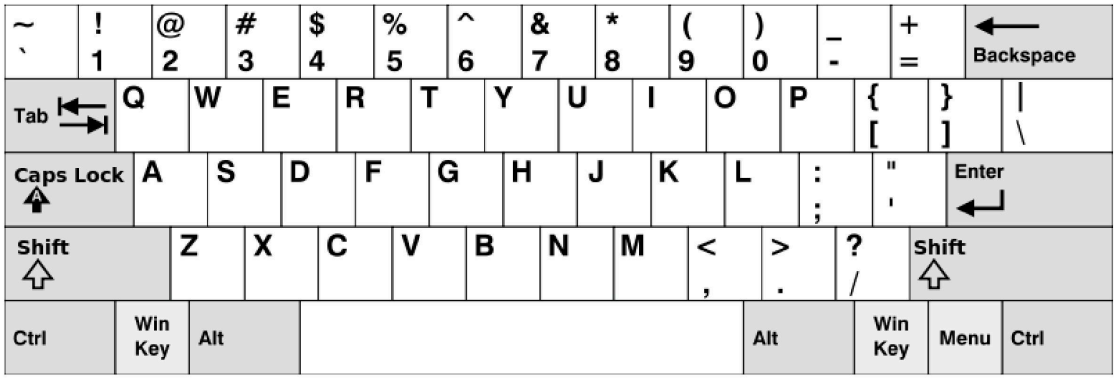
Register No.: **Name:**

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.

```
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))
```



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())
b=[list(input()) for x in range(a)]
c=[0,1,2,3,4,5,6,7,8,9,'!','@','#','$',"%",'^','&','*','(",")"]
d=['q','w','e','r','t','y','u','i','o','p']
e=['a','s','d','f','g','h','j','k','l','A','S','D','F','G','H','J','K','L',':',';',"'",'"']
f=['z','x','c','v','b','n','m',',','.','/',"\\",'?','|']
h=1
for i in b:
    g=0
    for j in i:
        if j in c:
            g+=1
        if j in d:
            g+=2
        if j in e:
            g+=3
    if(g==len(i) or g==len(i)*2 or g==len(i)*3):
        for k in i:
            print(k,end="")
        print("")
        h=0
if h:
    print("No words")
```