08 – Tuple/Set

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.

```
a=input()
cnt=0
for i in range(len(a)):
    if a[i]=='0' or a[i]=='1':
        cnt+=1
if cnt==len(a):
    print("Yes")
else:
    print('No')
```

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**.

```
s=input()
d=len(s)
l=m=10
count=0
b=[]
for i in range(d):
    for j in range(d):
        if s[i:l]==s[j+i:m]:
            count+=1
        m+=1
    if(count>1):
        b.append(s[i:l])
    count=0
    l+=1
    m=l

c=set(b)
for i in c:
```



```
    print(i)
```

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.

Solution:

```
nums=input().split()
for i in nums:
if nums.count(i)>1:
print(i)
break
```

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.

```
size1,size2=map(int,input().split())

array1=list(map(int,input().split()))

array2=list(map(int,input().split()))

set1=set(array1)

set2=set(array2)

non_repeating_elements=(set1.symmetric_difference(set2))

if non_repeating_elements:

    print(*sorted(non_repeating_elements))

    print(len(non_repeating_elements))

else:

    print("NO SUCH ELEMENTS")
```

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)
```

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".

```
def find_words(words):
    row1=set("qwertyuiop")
    row2=set("asdfghjkl")
    row3=set("zxcvbnm")
    def row(word):
        lower_word=set(word.lower())
        return lower_word<=row1 or lower_word<=row2 or lower_word<=row3
    return [word for word in words if row(word)]
import sys
input=sys.stdin.read
data=input().strip().split()
index=0
while index<len(data):
    n=int(data[index])
    index+=1
```