**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.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

**For example:**

| **Input** | **Result** |
| --- | --- |
| 01010101010 | Yes |
| 010101 10101 | No |

#Binary

s=input()

c=0

for i in s:

if i=='0' or i=='1':

c=c+1

if c==len(s):

print("Yes")

else:

print("No")

**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**.

**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 |

#Distinct pair

s=input()

k=int(input())

z=s.split(',')

l=[]

for i in range(0,len(z)):

for j in range(i+1,len(z)):

if int(z[i])+int(z[j])==k and [z[i],z[j]] not in l and [z[j],z[i]] not in l:

l.append([z[i],z[j]])

print(len(l))

**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**.

**Example 1:**

**Input:** s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

**Output:** ["AAAAACCCCC","CCCCCAAAAA"]

**Example 2:**

**Input:** s = "AAAAAAAAAAAAA"

**Output:** ["AAAAAAAAAA"]

**For example:**

| **Input** | **Result** |
| --- | --- |
| AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT | AAAAACCCCC  CCCCCAAAAA |

#DNA SQUENCE

s=input()

l=len(s)

w=[]

c=0

a=0

b=10

for i in range(0,len(s)-9):

s1=s[a:b]

if s1 in w and w.count(s1)==1:

print(s1,end='\n')

w.append(s1)

a=a+1

b=b+1

**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](http://118.185.187.137/moodle/mod/resource/view.php?id=734).

**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 |

 def find\_duplicate(nums):

seen = set()

for num in nums:

if num in seen:

return num

seen.add(num)

return -1

nums1 = input().split()

nums1=[int(i) for i in nums1]

print(find\_duplicate(nums1))

**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.

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Input:

5 4

1 2 8 6 5

2 6 8 10

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output:

1 5 10

3

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127)  Input:

5 5

1 2 3 4 5

1 2 3 4 5

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output:

NO SUCH ELEMENTS

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 4  1 2 8 6 5  2 6 8 10 | 1 5 10  3 |

#non repeating

a=input()

b=input()

c=input()

z1=b.split()

z2=c.split()

z=z1+z2

#print(z)

d=0

l=[]

for i in z:

c=0

if i in z2 and i in z1:

c=1

if c==0 and i not in l:

print(i,end=' ')

l.append(i)

d=d+1

if len(l)==0:

print("NO SUCH ELEMENTS")

print()

print(d)

**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.

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 |

#Keyboard

s1=input()

s=s1.lower()

a=list(input())

z=s.split()

d=0

for i in z:

c=0

for j in i:

if j in a:

c=1

break

if(c==0):

d=d+1

print(d)

**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".



**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 |

#american keyboard

kbRows = "qwertyuiop", "asdfghjkl", "zxcvbnm"

inList, outList = [input() for \_ in range(int(input()))],[]

for word in inList:

for row in kbRows:

if set(word.lower()).issubset(set(row)):

outList.append(word)

if outList : print(\*outList, sep='\n'); exit();

print('No words')