

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.

```
t = set(map(int,input().split(",")))
k = int(input())
t = list(t)
count = []
for i in range(len(t)):
    for j in range(i,len(t)):
        if t[i] + t[j] == k:
            temp = (t[i],t[j])
            count.append(temp)

print(len(count))
```

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**.

```
t = set(map(int,input().split(",")))
k = int(input())
t = list(t)
count = []
for i in range(len(t)):
    for j in range(i,len(t)):
        if t[i] + t[j] == k:
            temp = (t[i],t[j])
            count.append(temp)

print(len(count))
```

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC","CCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAAA"

Output: ["AAAAAAAAA"]

For example:

Input	Result
AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC CCCCAAAAA

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](#).

```
a=[int(i) for i in input().split()]
n1=[int(i) for i in input().split()]
n2=[int(i) for i in input().split()]
com=set(n1)&set(n2)
uni=set(n1) | set(n2)
tot=sorted(uni-com)
for i in tot:
    print(i,end=' ')
print('\n',end='')
print(len(tot))
```

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.

```
a=[int(i) for i in input().split()]
```

```
n1=[int(i) for i in input().split()]
```

```
n2=[int(i) for i in input().split()]
```

```
com=set(n1)&set(n2)
```

```
uni=set(n1) | set(n2)
```

```
tot=sorted(uni-com)
```

```
for i in tot:
```

```
    print(i,end=' ')
```

```
print('\n',end="")
```

```
print(len(tot))
```

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))
```

~ `	!	@	#	\$	%	^	&	*	()	-	+	Backspace
Tab	Q	W	E	R	T	Y	U	I	O	P	{	}	
Caps Lock	A	S	D	F	G	H	J	K	L	:	"	Enter	
Shift	Z	X	C	V	B	N	M	<	>	?	Shift		
Ctrl	Win Key	Alt									Alt	Win Key	Menu Ctrl

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".

```
n=int(input())
f=0
a=[input() for i in range(n)]
l1=['qwertyuiop','asdfghjkl','zxcvbnm']
l=[[j for j in i] for i in l1]
for i in a:
    n=[j for j in i.lower()]
    #print(sorted(set(l[1]) | set(n))==sorted(set(l[1])))
    #print(set(l[1]),set(n))
    if set(n) | set(l[0])==set(l[0]):
        f=1
        print(i)
        continue
    elif set(n) | set(l[1])==set(l[1]):
```



```
f=1  
  
print(i)  
  
continue  
  
elif set(n) | set(l[2])==set(l[2]):  
  
    f=1  
  
    print(i)  
  
    continue  
  
if not f:  
  
    print('No words')
```