1. 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.

**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 |

**Program:**

def findDuplicate(nums):

seen = set()

for num in nums:

if num in seen:

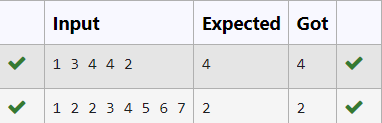
return num

seen.add(num)

user\_list = [int(x) for x in input().split()]

print(findDuplicate(user\_list))

**Output:**

****

2. 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

**Program:**

text=input()

brok=input()

c=0

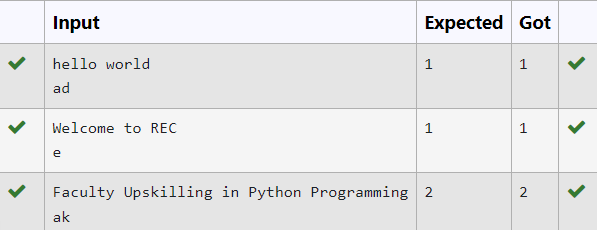
for i in brok:

if i in text:

c=c+1

print(c)

**Output:**

****

3. 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 |

Program:

def findRepeatedSequences(s):

sequences = {}

result = []

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

seq = s[i:i+10]

sequences[seq] = sequences.get(seq, 0) + 1

if sequences[seq] == 2:

result.append(seq)

return result

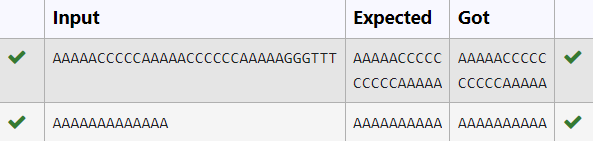
# Example usage

s1 = input()

for i in findRepeatedSequences(s1):

print(i)

**Output:**

****

4. 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 |

**Program:**

def count\_distinct\_pairs(t, K):

distinct\_pairs = set()

for i in range(len(t)):

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

if t[i] + t[j] == K:

distinct\_pairs.add((min(t[i], t[j]), max(t[i], t[j])))

return len(distinct\_pairs)

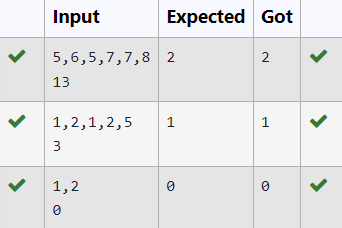
t\_input = input()

t = tuple(map(int, t\_input.split(',')))

K = int(input())

print(count\_distinct\_pairs(t, K))

**Output:**

****

**5.** 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 |
| 2  adsfd  afd | adsfd  afd |

**Program:**

def findWords(words):

row1 = set('qwertyuiop')

row2 = set('asdfghjkl')

row3 = set('zxcvbnm')

result = []

for word in words:

w = set(word.lower())

if w.issubset(row1) or w.issubset(row2) or w.issubset(row3):

result.append(word)

if len(result) == 0:

print("No words")

else:

for i in result:

print(i)

a = int(input())

arr = [input() for i in range(a)]

findWords(arr)

**Output:**

****