```
    print(len(n))
else:
    print("NO SUCH ELEMENTS")
```

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.: 230701363 **Name: THEEPAK PALANI KUMAR D**

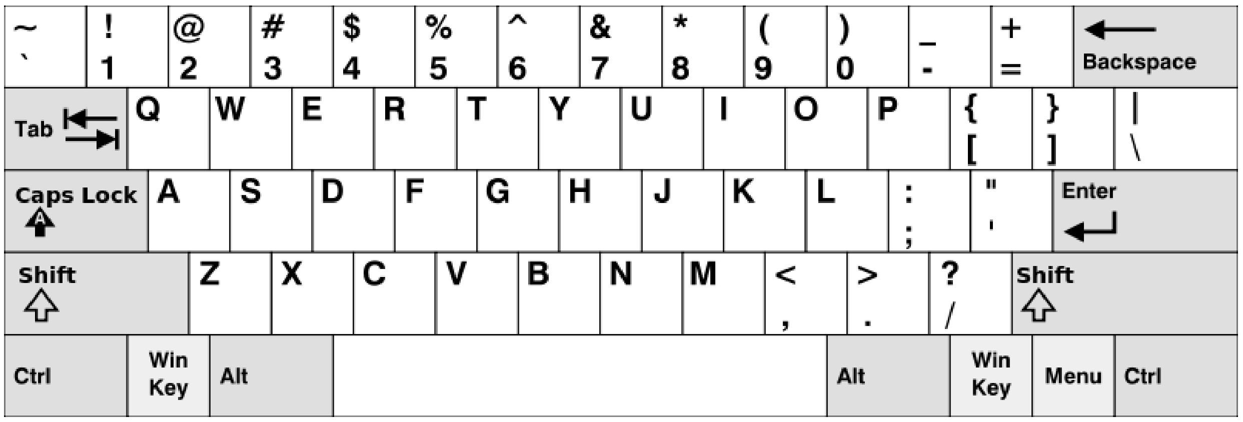
Malfunctioning Keyboard

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.

Program:

```
n=input()
b=n.lower()
a=input()
word=b.split()
count=0
for i in word:
    if any(letter in a for letter in i):
        continue
    else:
        count+=1
print(count)
```



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.: 230701363 **Name: THEEPAK PALANI KUMAR D**

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".

Program:

```
key=['qwertyuiop', 'asdfghjkl','zxcvbnm']

n=int(input())

words=[]

results=[]

for i in range(n):

    a=input()

    words.append(a)

for word in words:

    lower= word.lower()

    found=False

    for row in key:

        if all(letter in row for letter in lower):

            found=True

            break
    if found:
        results.append(word)
if(len(results)==0):
    print("No words")
else:
    for i in results:
        print(i)
```

09 – Dictionary

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use dictionary to solve the problem

For example:

Input	**Result**
this apple is sweet this apple is sour	sweet sour

Ex. No. : 9.1 **Date:**

Register No.: 230701363 **Name: THEEPAK PALANI KUMAR D**

Uncommon words

A sentence is a string of single-space separated words where each word consists only of lowercase letters.A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

Program:

```
s1 = input()
s2 = input()
words_s1 = {word: s1.split().count(word) for word in set(s1.split())}
words_s2 = {word: s2.split().count(word) for word in set(s2.split())}
uncommon_words = [word for word in words_s1 if words_s1[word] == 1 and word not in words_s2] + \
            [word for word in words_s2 if words_s2[word] == 1 and word not in words_s1]
print(' '.join(uncommon_words))
```

Input : test_dict = {‘Gfg’ : [6, 7, 4], ‘best’ : [7, 6, 5]}

Output : {‘Gfg’: 17, ‘best’: 18}

Explanation : Sorted by sum, and replaced.

Input : test_dict = {‘Gfg’ : [8,8], ‘best’ : [5,5]}

Output : {‘best’: 10, ‘Gfg’: 16}

Explanation : Sorted by sum, and replaced.

Sample Input:

2

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

For example:

Input	**Result**
2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18