Started on	Saturday, 1 June 2024, 11:48 PM
State	Finished
Completed on	Saturday, 1 June 2024, 11:54 PM
Time taken	5 mins 41 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

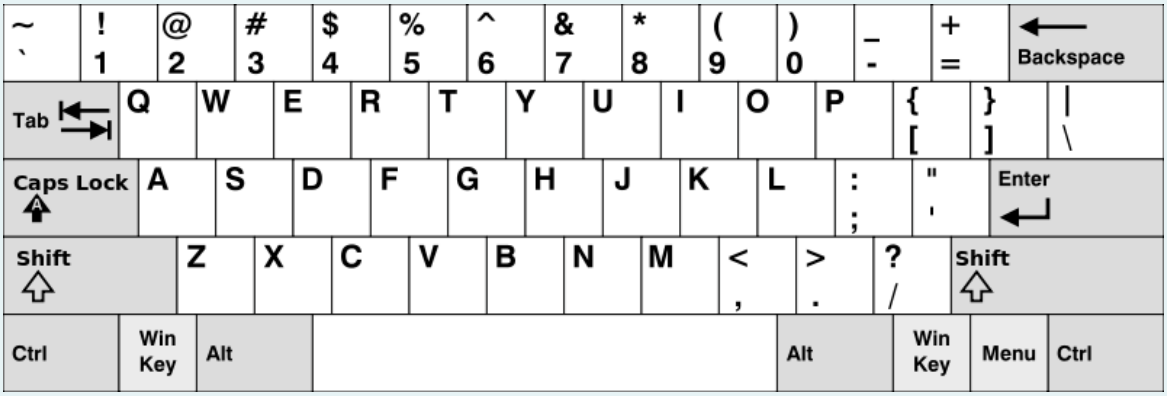
Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```
def findWords(words):
  """
  :type words: List[str]
  :rtype: List[str]
  """
  rows = ["qwertyuiop", "asdfghjkl", "zxcvbnm"]
  result = []
  for word in words:
    row_found = False
    for row in rows:
      if set(word.lower()) <= set(row):
        row_found = True
```

```
        break
    if row_found:
      result.append(word)
  return result

# Get user input for words (without instructions)
num_words = int(input())
words = []
for _ in range(num_words):
  word = input()
  words.append(word)

# Find words on one row
one_row_words = findWords(words)

# Print results with empty set handling
if not one_row_words:
  print("No words")
else:
  print("\n".join(one_row_words))
```

	Input	Expected	Got	
✓	4 Hello Alaska Dad Peace	Alaska Dad	Alaska Dad	✓
✓	1 omk	No words	No words	✓
✓	2 adsfd afd	adsfd afd	adsfd afd	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```
s = input()
A = set()
B = set()
for i in range(len(s) - 9):
    C = s[i:i + 10]
    if C in A:
        B.add(C)
    else:
        A.add(C)
for seq in B:
    print(seq)
```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA	AAAAACCCCC CCCCCAAAAA	✓
✓	AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

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 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
5 5 1 2 3 4 5 1 2 3 4 5	NO SUCH ELEMENTS

Answer: (penalty regime: 0 %)

```
def non_repeating_elements(arr1, arr2):
    set1 = set(arr1)
    set2 = set(arr2)
    non_repeating = set1.symmetric_difference(set2)
    if not non_repeating:
        print("NO SUCH ELEMENTS")
    else:
        print(*non_repeating)
        print(len(non_repeating))

# Read input sizes and arrays
size1, size2 = map(int, input().split())
arr1 = list(map(int, input().split()))
arr2 = list(map(int, input().split()))

# Call the function
non_repeating_elements(arr1, arr2)

```

	Input	Expected	Got	
✓	5 4 1 2 8 6 5 2 6 8 10	1 5 10 3	1 5 10 3	✓
✓	3 3 10 10 10 10 11 12	11 12 2	11 12 2	✓
✓	5 5 1 2 3 4 5 1 2 3 4 5	NO SUCH ELEMENTS	NO SUCH ELEMENTS	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```
nums = list(map(int, input().split()))

seen =[]
for num in nums:
    if num in seen:
        print(num)
        break
    seen.append(num)

```

	Input	Expected	Got	
✓	1 3 4 4 2	4	4	✓
✓	1 2 2 3 4 5 6 7	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```
t = tuple(map(int, input().split(',')))
K = int(input())

seen = {}
distinct_pairs = set()

for num in t:
    complement = K - num
    if complement in seen and seen[complement] > 0:
        distinct_pairs.add((min(num, complement), max(num, complement)))
        seen[complement] -= 1
    else:
        seen[num] = seen.get(num, 0) + 1

print(len(distinct_pairs))
```

	Input	Expected	Got	
✓	5,6,5,7,7,8 13	2	2	✓
✓	1,2,1,2,5 3	1	1	✓
✓	1,2 0	0	0	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.