

Examples:

Input: str = "0101010101010"

Output: Yes

Input: str = "REC101"

Output: No

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.

```
a=input()
c=0
for i in range(len(a)):
    if (a[i]=='0' or a[i]=='1'):
        c=c+1

if c==len(a):
    print("Yes")
else:
    print("No")
```

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.

Input	Result
1,2,1,2,5	1
1,2	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.

```
tuple_str = input()
t = tuple(map(int, tuple_str.split(",")))
K = int(input())

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])))
print(len(distinct_pairs))
```

Input: s = "AAAAACCCCCCAAAAACCCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAA"

Output: ["AAAAAAAAAA"]

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA

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**.

```
s=input()
if len(s)<=10:
    print()
else:
    sequences={}
    result=[]
    for i in range(len(s)-9):
        sequence=s[i:i+10]
        if sequence in sequences and sequence not in result:
            result.append(sequence)
        else:
            sequences[sequence]=True
    for seq in result:
        print(seq)</pre>
```

Input: nums = [1,3,4,2,2]

Output: 2

Example 2:

Input: nums = [3,1,3,4,2]

Output: 3

Input	Result
1 3 4 4 2	4

Ex. No.	:	8.4	Date:		
Register No.:			Name:	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 <u>set</u>.

a = input()

b = a.split() # Call split as a function

seen = set()

for num in b:

if num in seen:

print(num)

else:
seen.add(num)

Sample Input:

5 4

 $1\,2\,8\,6\,5$

26810

Sample Output:

 $15\ 10$

3

Sample Input:

5 5

 $1\ 2\ 3\ 4\ 5$

 $1\ 2\ 3\ 4\ 5$

Sample Output:

NO SUCH ELEMENTS

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 = input().split()
n1 = input().split()
n2 = input().split()
c = set()
for num in n1:
  if num not in n2:
     c.add(num)
for num1 in n2:
  if num1 not in n1:
     c.add(num1)
c = sorted(map(int, c))
if c:
 print(" ".join(map(str, c)))
print(len(c))
else:
print("NO SUCH ELEMENTS")
```

Input: text = "hello world", brokenLetters = "ad"

Output:

1

Explanation: We cannot type "world" because the 'd' key is broken.

Input	Result
hello world ad	1

Ex. No. : 8.6 Date:

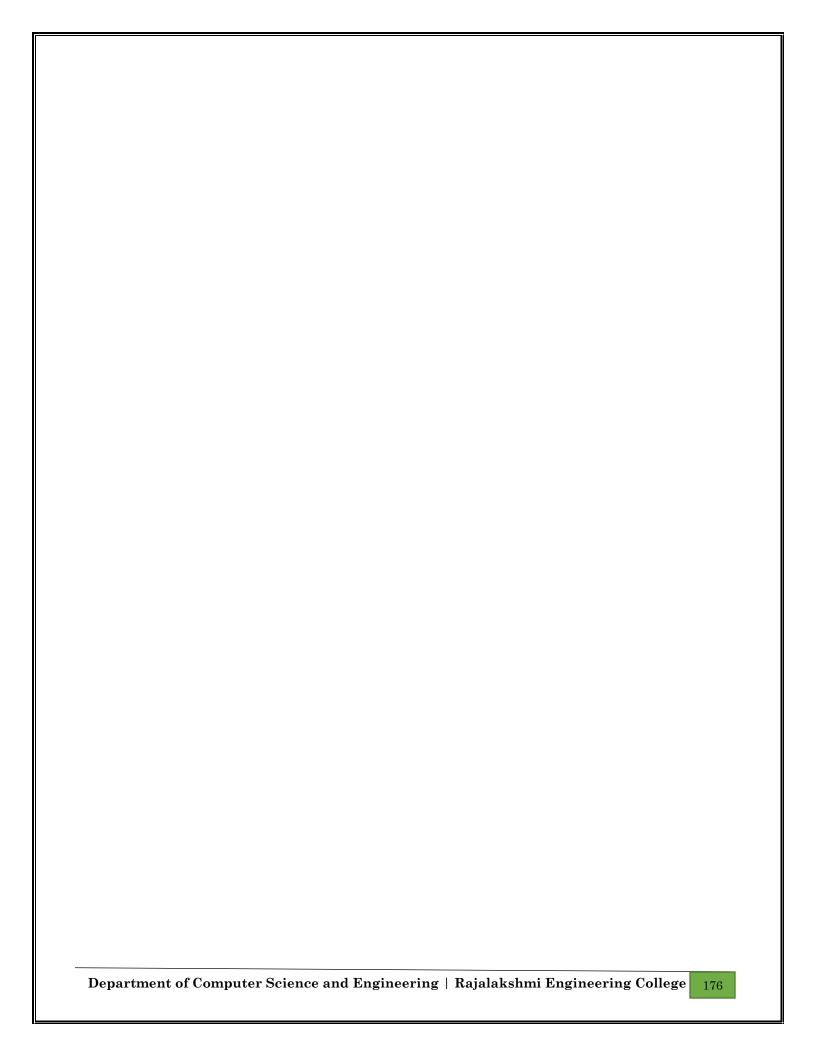
Register No.: Name:

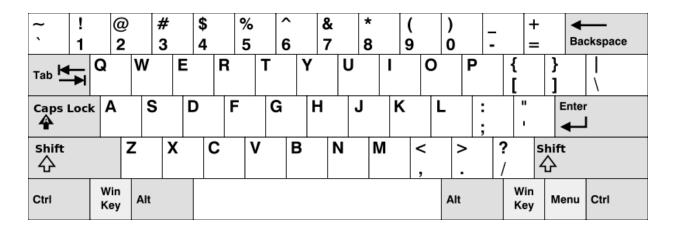
Malfunctioning Keyboard

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.

```
def can_type_word(word, broken_letters):
  for letter in word:
    if letter in broken letters:
      return False
  return True
def fully_typed_words(text, broken_letters):
  words = text.split()
  fully typed count = 0
  for word in words:
    if can_type_word(word, broken_letters):
      fully_typed_count += 1
 return fully_typed_count
```





Input: words = ["Hello","Alaska","Dad","Peace"]

Output: ["Alaska","Dad"]

Example 2:

Input: words = ["omk"]

Output: [] Example 3:

Input: words = ["adsdf","sfd"]
Output: ["adsdf","sfd"]

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 "gwertyuiop",
- the second row consists of the characters "asdfghjkl", and
- the third row consists of the characters "zxcvbnm".

```
rows=["qwertyuiop","asdfghjkl","zxcvbnm"]
n=int(input())
words=[]
for i in range(n):
    word=input()
    words.append(word)
l=[]
for word in words:
    flag=False
    for row in rows:
        if all(char in row for char in word.lower()):
            flag=True
            break
        if flag:
```

```
l.append(word)

if l:
   for i in range(len(l)):
      print(l[i])

else:
   print("No words")
```

