WEEK 7-PYTHON PROGRAMMING

1.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.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

Input	Result
01010101010	Yes
010101 10101	No

```
SOLUTION:
a=input()
d=0
for i in a:
    if i=='0' or i=='1':
        d=2
    else:
        d=1
        break

if d==2:
    print('Yes')
elif d==1:
    print('No')
else:
    print(0)
```

2. 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:

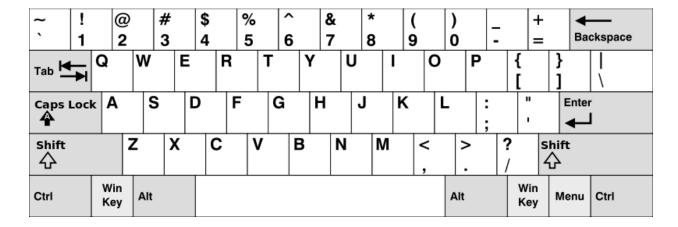
```
Input: t = (5, 6, 5, 7, 7, 8), K = 13
Output: 2
Explanation:
Pairs with sum K(=13) are \{(5, 8), (6, 7), (6, 7)\}.
Therefore, distinct pairs with sum K(=13) are \{(5, 8), (6, 7)\}.
Therefore, the required output is 2.
```

Input	Result
1,2,1,2, 5	1
1,2	0

```
SOLUTION:
a=input()
li1=eval(a)
#print(li1)
d=len(li1)
#print(d)
k=int(input())
li=[]
for i in range(0,d):
  for j in range(i,d):
     if li1[i]+li1[j]==k:
        li.append((li1[i],li1[j]))
#print(li)
li=set(li)
#print(li)
p=len(li)
print(p)
```

3. 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".



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"]
```

Input	Result
4	Alask
Hello	a
Alask	Dad
a	
Dad	

Peace	
2 adsfd afd	adsfd afd

```
SOLUTION:
a=int(input())
lst=[]
for i in range(0,a):
   b=input("")
   lst.append(b)
lst2=['q','w','e','r','t','y','u','i','o','p','Q','W','E','R','T','Y','U','l','O','P']
lst3=['a','s','d','f','g','h','j','k','l','A','S','D','F','G','H','J','K','L']
lst4=['z','x','c','v','b','n','m','Z','X','C','V','B','N','M']
I=0
m=0
n=0
lst5=[]
for i in lst:
   I=0
   m=0
   n=0
  j=i
   b=len(j)
  for k in range(0,b):
     if(i[k] not in lst3 and i[k] not in lst4):
     elif(i[k] not in lst2 and i[k] not in lst4):
     elif(i[k] not in lst2 and i[k] not in lst3):
        n+=1
   if(l==b or m==b):
     lst5.append(i)
p=0
for i in lst5:
   p+=1
   if i!=0:
     print(i)
if(p==0):
   print("No words")
```

- 4. 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 = "AAAAAAAAAAAA"
Output: ["AAAAAAAAAAA"]
```

Input	Result
AAAAACCCCCAAAAACCCCCCAAAAAGGG TTT	AAAAACCCC C CCCCCAAAA A

```
Answer:(penalty regime: 0 %)

SOLUTION:
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)

5. 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

```
SOLUTION:

def find_duplicate(nums):
    seen = set()
    for num in nums:
        if num in seen:
        return num
        seen.add(num)
```

Input

```
I = input()
nums = list(map(int, I.split()))
# Output
print(find_duplicate(nums))
```