Even or Odd String

A string is a palindrome if and only if it reads the same forwards and backwards.

A substring of a string **s** is a nonempty sequence of consecutive characters from **s**

Define a string to be odd if and only if all substrings of the string that are palindromes have odd length.

Given a string **s**, determine if it is odd.

Print on a single line, if the string is odd, Odd; if not, Even.

**Sample Input:**

Amanaplanacanalpanama

**Sample Output:**

Odd

**Sample Input:**

Madamimadam

**Sample Output:**

Odd

**Sample Input:**

Annamyfriend

**Sample Output:**

Even

Longest Substring without Repeating Characters:

Given a string, find the length of the **longest substring** without repeating characters.

**Sample Input:**

abcabcbb

**Sample Output:**

3

**Sample Input:**

bbbbb

**Sample Output:**

1

**Sample Input:**

pwwkew

**Sample Output:**

3

def lengthOfLongestSubstring(s):

"""

:type s: str

:rtype: int

"""

ans = 0

sub = ''

for char in s:

if char not in sub:

sub += char

ans = max(ans, len(sub))

else:

cut = sub.index(char)

sub = sub[cut+1:] + char

return ans

**Permutation in String**

Given two strings **s1** and **s2**, write a function to return true if **s2** contains the permutation of **s1**. In other words, one of the first string's permutations is the **substring** of the second string.

**Sample Input:**

s1 = "ab" s2 = "eidbaooo"

**Sample Output:**

True

**Sample Input:**

s1= "ab" s2 = "eidboaoo"

**Sample Output:**

False

def checkInclusion(s1, s2):

if (len(s1) > len(s2)):

return false;

s1map = [0] \* 26

for i in range(len(s1)):

s1map[ord(s1[i]) - ord('a')] += 1;

#print("s1",s1map)

for i in range(len(s2) - len(s1)+1):

s2map = [0] \* 26;

for j in range(len(s1)):

s2map[ord(s2[i+j]) - ord('a')] += 1;

#print("s2:",s2map)

if (matches(s1map, s2map)):

return True

return False

def matches(s1map, s2map):

for i in range(26):

if (s1map[i] != s2map[i]):

return False

return True

sol = checkInclusion("ab","eidboaoo")

print(sol)

**First Unique Character in a String**

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

**Sample Input:**

s = "liveshow"

**Sample Output:**

0

**Sample Input:**

s = "loveliveshow"

**Sample Output:**

5

def firstUniqChar(s):

count = [0] \* 26

# find the index

for i in range(len(s)):

count[ord(s[i]) - ord('a')] += 1;

index = 0

for ch in s:

if count[ord(ch) - ord('a')] == 1:

return index

else:

index += 1

return -1

**Backspace String Compare**

Given two strings S and T, return if they are equal when both are typed into empty text editors. # means a backspace character.

**Sample Input:**

S = "ab#c", T = "ad#c"

**Sample Output:**

True

**Sample Input:**

S = "ab##", T = "c#d#"

**Sample Output:**

True

**Sample Input:**

S = "a#c", T = "b"

**Sample Output:**

False

def backspaceCompare(S, T):

return build(S) == build(T)

def build(S):

ans = []

for c in S:

if c != '#':

ans.append(c)

elif ans:

ans.pop()

return "".join(ans)