Count Substring of 0 and 1

Given a binary string **s**, return the number of **non-empty** substrings that have the same number of **0's** and **1's**, and all the **0's** and all the **1's** in these substrings are grouped consecutively. Substrings that occur multiple times are counted the number of times they occur.

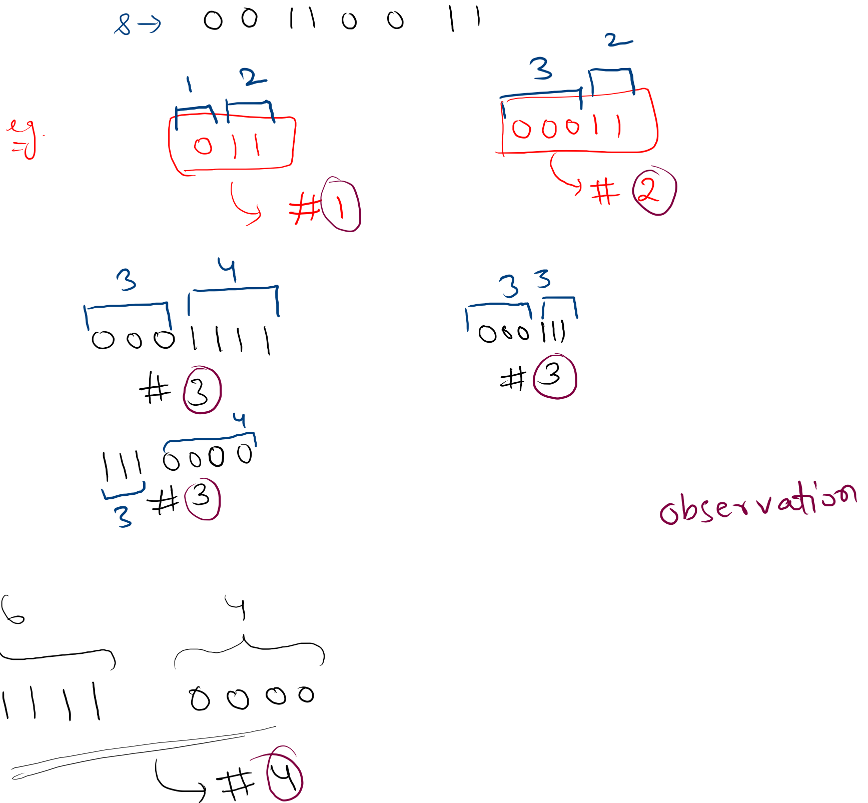
NOTE:- After answering the question, attempt the related question in the linked resource to improve your understanding of this question .Click here

Sample Input 0

```
00110011
```

Sample Output 0

```
6
```



$$\underbrace{\quad}_{0}\ \underbrace{000}_{3}\ \underbrace{11}_{2}\ \underbrace{000}_{3}\ \underbrace{1}_{1}\ \underbrace{000}_{3}$$

$$0 + \boxed{2 + 2 + 1 + 1} = 6$$

$$\underbrace{0000}_{4}\ \underbrace{11}_{2}\ \underbrace{000}_{3}$$

$$\boxed{\boxed{2} + 2}$$

prev = ~~0~~ ~~3~~ ~~2~~ ~~3~~ 1

curr = ~~1~~ ~~2~~ ~~1~~ ~~2~~ ~~3~~ ~~1~~ ~~2~~ 3

ans = ~~0~~ ~~2~~ ~~4~~ ~~5~~ 6

$$0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ \underbrace{1}_{p}\ \underbrace{0\ 0\ 0}_{c}\ \Big\}\ i$$

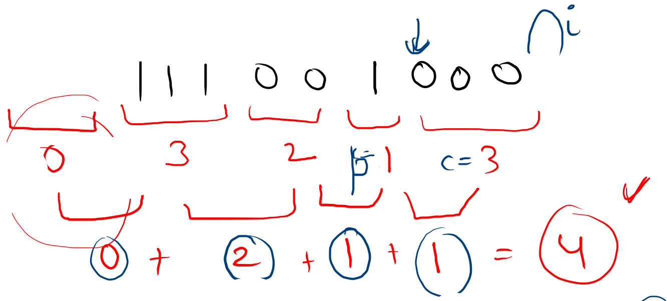
```
if (A[i] == A[i-1])
    ↳ curr++

else {
    ans += min(prev, curr)
    prev = curr
    curr = 1
}
```

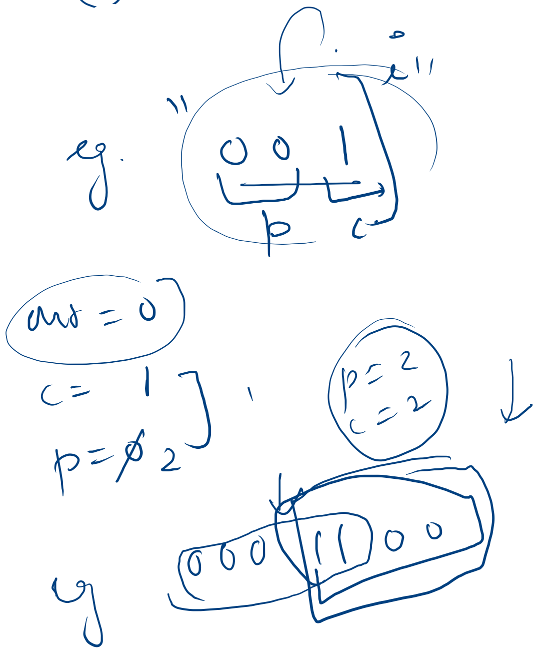
$ans = \cancel{0}\ \cancel{2}\ \cancel{3}\ 4$

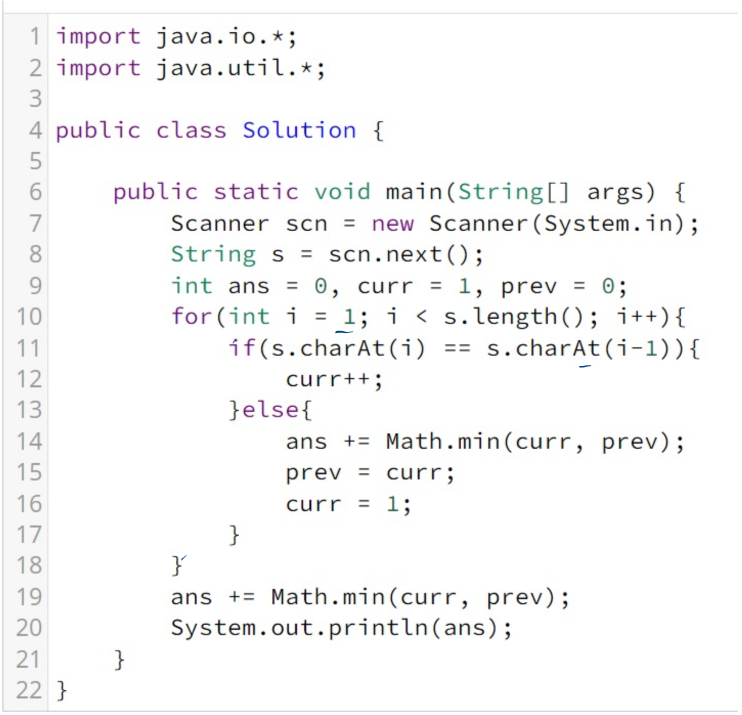
$curr = \cancel{1}\ \cancel{2}\ 3$

$prev = 1$

$ans = 1$





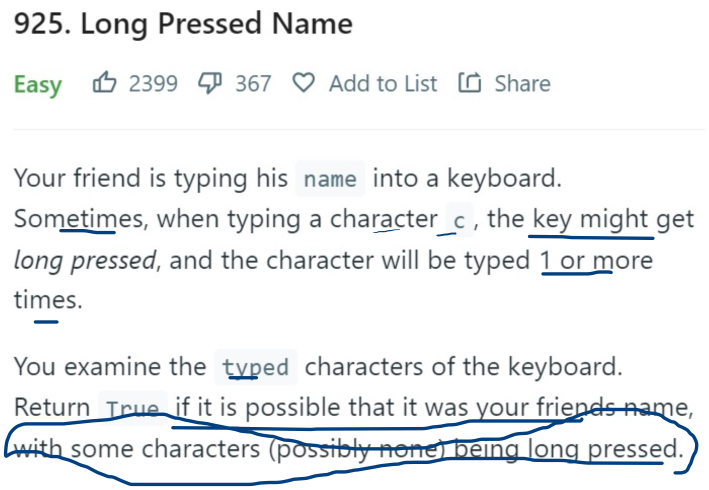


```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        int ans = 0, curr = 1, prev = 0;
        for(int i = 1; i < s.length(); i++){
            if(s.charAt(i) == s.charAt(i-1)){
                curr++;
            }else{
                ans += Math.min(curr, prev);
                prev = curr;
                curr = 1;
            }
        }
        ans += Math.min(curr, prev);
        System.out.println(ans);
    }
}
```

Long Pressed Name

Your friend is typing his name into a keyboard. Sometimes, when typing a character **c**, the key might get **long pressed**, and the character will be typed 1 or more times.

You examine the typed characters of the keyboard. Return **True** if it is possible that it was your friends name, with some characters (possibly none) being long pressed.



925. Long Pressed Name

Easy 👍 2399 👎 367 ♡ Add to List Share

Your friend is typing his `name` into a keyboard. Sometimes, when typing a character `c`, the key might get *long pressed*, and the character will be typed 1 or more times.

You examine the `typed` characters of the keyboard. Return `True` if it is possible that it was your friends name, with some characters (possibly none) being long pressed.

want to

name ↳ "aman"

actually

typed → aaamannn

aman ← aaaaman → T

aman bbmaan → False

eg1. n→ alex

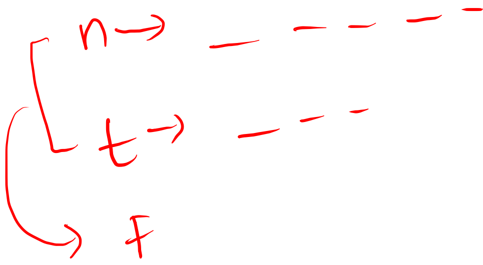
t→ allex

#T

eg2. n→ allex

t→ alex

F



eg3. n→ alex

t→ apex

#F

eg4. alex

alex

#T

eg5. n→ a$\boxed{\text{lll}}$ex

t→ allex

#F

eg6. n→ alllex

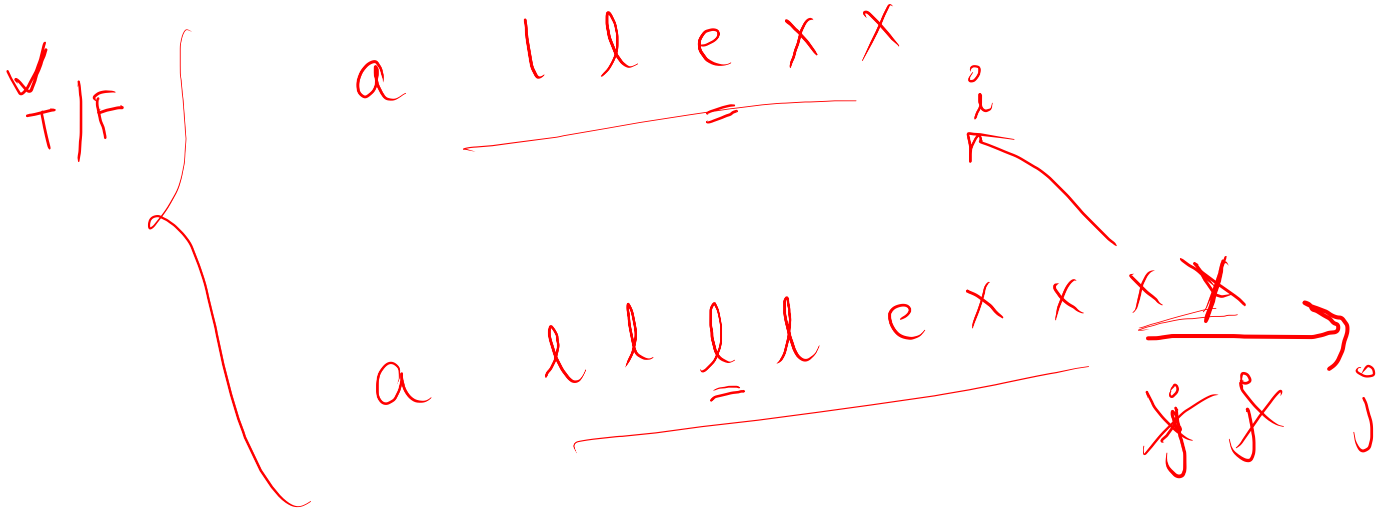
t→ allexX

#F

eg7. n→ allexx

t→ alllexxxx

#T

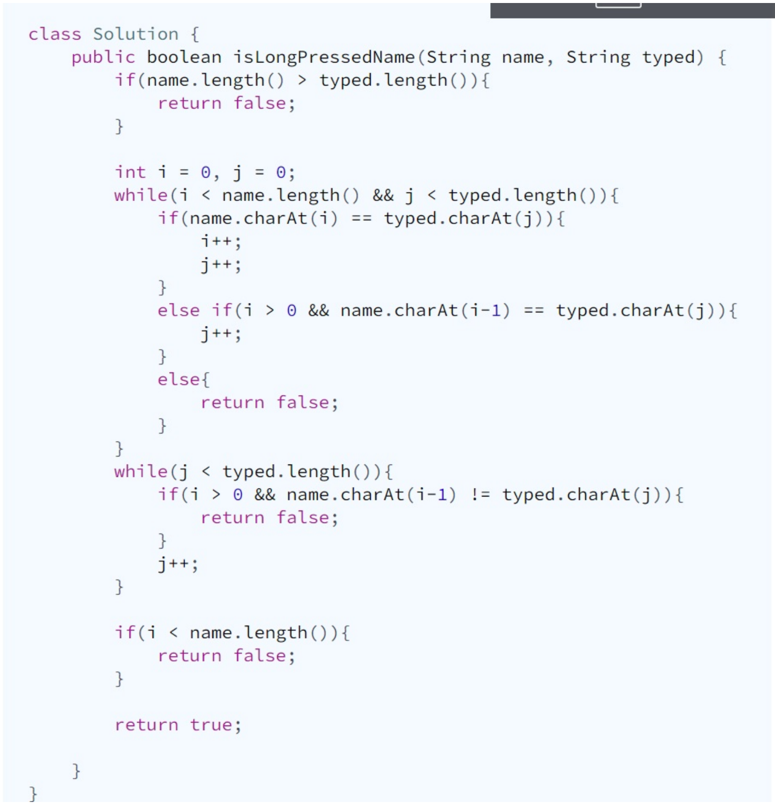

✓
T|F
a l l e x x
i
a l l l l e x x x x
j
j
j

eg. → n → a l e x x t

i

a l e x X x

j



```java
class Solution {
    public boolean isLongPressedName(String name, String typed) {
        if(name.length() > typed.length()){
            return false;
        }

        int i = 0, j = 0;
        while(i < name.length() && j < typed.length()){
            if(name.charAt(i) == typed.charAt(j)){
                i++;
                j++;
            }
            else if(i > 0 && name.charAt(i-1) == typed.charAt(j)){
                j++;
            }
            else{
                return false;
            }
        }
        while(j < typed.length()){
            if(i > 0 && name.charAt(i-1) != typed.charAt(j)){
                return false;
            }
            j++;
        }

        if(i < name.length()){
            return false;
        }

        return true;

    }
}
```