Minimum characters to be added at front to make string palindrome

we need to check a string it is a palindrome or not, if it is a palindrome then return 0, else we need to add some characters to the front of string such that characters added would be minimum as possible.

For example,

consider string,

S = "abc"

i=0
j=2
last_idx=2

→ check
↓ s[0:2]
Not passed

i=0
last_idx--
j=last_idx

step 2:

i=0 j=1 last_idx=1

→ Check { S="abc"
↓
Not palin s[0]=s[i]
→ i=0
last_idx--
j=last_idx

step 3:

i=0 j=0 last_idx=0

→ check { S="abc"
↓
palin s[0]=s[0]
→ i=1 j=0

Now the Min length would be

n-1-last_index

→ 3-1-0 ⇒ 2

we do, return n-1-last_index because, last_indx counts num of mismatches which are avoiding the current string to become a palindrome

Algorithm:

we need to iterate through the string until i<=j, we need to stop when it becomes i>j

Now we need to check, if str[i] == str[j], initially i=0 & j=n-1, if it is true, then do i++ & j--

If str[i] ≠ str[j] then, we can assume, we need to add str[j] to start of the string. so, neglect that index for now. Then do,

i=0
last_index--
j=last_index

Initially, last_index is pointing to n-1

```
while (i<=j){
    if (str[i] == str[j]){
        i++
        j--
    }else{
        i=0 & last_index--;
        j=last_index
    }
}
return (n-1 - last_index)
```

else case is used to count these mismatches

⇓

which returns, minimum characters to be added at front to make string palindrome.