Check Palindrome (recursive)

Problem Description: Check if a given String S is palindrome or not (using recursion). Return true or false.

Sample Input:

```
racecar
```

Sample Output:

```
true
```

How to approach?

A palindrome string is one that is the same whether it's read left-to-right or right-to-left. Some examples of palindromic strings are, "malayalam", "racecar", "a". One way to look at palindromes is that they are equal to their reverse. So, we could reverse a string and then compare it with itself. If they are equal, then the string is a palindrome, otherwise it's not.

Another way to look at them is that the first character must be the same as the last character, the second character must be the same as the second last character and so on.

Programmatically speaking, the second way of looking at palindromes is better since we won't have to reverse the string and store it somewhere and then compare the two character by character. We could just use two pointers (one from the 0th index and one from the last index) to iterate through the string and compare the characters, we they are equal, we move to the next iteration, otherwise we straight-away return false.
This is easier to do iteratively, but a little tricky to do recursively though.

One thing that we can do is to create a helper function and give it three parameters; the string `s` itself, the `start` index and the `end` index. Initially, the start would be 0, the end would be the last index of the string. Then we compare the characters at both these

indices. If they are equal, we recurse on our function passing the same string, `start + 1` and `end - 1`, otherwise we can just return false and skip further checking.
What will be the base case? Let’s look at the most trivial case. A string of length 0 or 1 is a palindrome. No question about it. So, how do know that we are dealing with a 0 or 1 size string? It’s easy. The start index must be equal to the end index (for string size 1) or it must be greater than the end index (for string size 0). So our base case could be:

```
if(start >= end):
   return true
```

The pseudo-code for this approach is shown on the next page.

```
function solve(str: string, start: integer, end: integer):
  if(start >= end):
    return true
  if(str[start] != str[end]):
    return false
  return solve(str, start + 1, end - 1);
```

```
function checkPalindrome(str: a string):
  return solve(str, 0, str.length() - 1);
```