Reverse String using Recursion

The provided C++ program is designed to reverse a given string using two different recursive functions. Both functions use the **swap** function from the **<algorithm>** header to exchange characters in the string.

Recursive approach to reverse a string

1. In the **reverseString** function with two parameters:
    - Base Case: If **start** is greater than or equal to **end**, the function returns without doing anything, as there is no need to swap characters further.
    - Recursive Case: The function swaps the characters at indices **start** and **end** using the **swap** function and then makes a recursive call to **reverseString** with **start + 1** and **end - 1**, effectively moving towards the center of the string.
2. In the **reverseString** function with one parameter:
    - Base Case: If **start** is greater than or equal to **str.length() - 1 - start**, the function returns without doing anything. This condition ensures that the function stops when it reaches the middle of the string.
    - Recursive Case: The function swaps the characters at indices **start** and **str.length() - 1 - start** using the **swap** function and then makes a recursive call to **reverseString** with **start + 1**, effectively moving towards the center of the string.

Time Complexity:

The time complexity of both **reverseString** functions is O(N/2), where N is the length of the string. This is because both functions process approximately half of the characters in the string due to the recursive approach. **However, since constants are usually dropped in Big O notation, we can simply say that the time complexity is O(N).**

Space Complexity:

The space complexity of both reverseString functions is O(N), where N is the length of the string. This is because the recursive calls create new frames on the call stack for each recursive call, and in the worst case, there can be N/2 recursive calls, leading to O(N) space consumption on the call stack.

Recursive call stack for the approach: