Check if String is Palindrome CodeStudio

The task is to determine whether a given string is a palindrome or not. A palindrome is a string that reads the same forwards and backwards, considering only alphanumeric characters and ignoring case.

Example: Input: "N2 i&nJA?a& jnI2n" Output: The String "N2 i&nJA?a& jnI2n" is a palindrome.

Approach 1: Function to check if a string is a palindrome

- The function first creates a reversed version of the input string using the **reverseString** function.
- It then iterates through the original and reversed strings simultaneously.
- Non-alphanumeric characters are skipped using **isalnum** function.
- Alphanumeric characters are compared while ignoring case using **tolower**.
- If a mismatch is found, the function returns **false**.
- If all characters match, the function returns **true**.

Time Complexity: The checkPalindrome function has a time complexity of O(n), where n is the length of the input string. This is because it involves iterating through the string once and performing comparisons.

Space Complexity: The checkPalindrome function creates an additional reversed string, which requires O(n) extra space, where n is the length of the input string. Therefore, the space complexity is O(n).

Approach 2: Optimized function to check if a string is a palindrome

- This function directly operates on the input string.
- It uses two pointers (**start** and **end**) that move towards each other from opposite ends of the string.
- Non-alphanumeric characters are skipped using **isalnum** function.
- Alphanumeric characters are compared while ignoring case using **tolower**.
- If a mismatch is found, the function returns **false**.
- If the pointers meet in the middle without any mismatches, the function returns **true**.

Time Complexity: The checkPalindromeOptimized function also has a time complexity of O(n), as it involves a single pass through the string with the two pointers.

Space Complexity: The checkPalindromeOptimized function does not require any additional space, except for a few variables. Hence, the space complexity is O(1).

Which approach to use?

- Both approaches have the same time complexity, but **checkPalindromeOptimized** has a better space complexity since it avoids creating a separate reversed string.
- Therefore, it is recommended to use the **checkPalindromeOptimized** function for palindrome checking, as it provides a more efficient solution.