Reverse String

The goal of this code is to reverse a given string using various approaches: iterative, recursive, iterators, and reverse iterators.
Input: "hello world",

Output: dlrow olleh

Approach 1: Function to reverse the string iteratively

- This function uses two pointers, **start** and **end**, starting from the first and last characters of the string.
- It iterates until **start** becomes greater than or equal to **end**.
- In each iteration, it swaps the characters at **start** and **end** indices and increments **start** and decrements **end**.
- This process continues until the middle of the string is reached, effectively reversing the string.
- **Time Complexity: O(n), where n is the length of the string. The function iterates through half of the string.**
- **Space Complexity: O(1), as the reversal is performed in-place.**

Approach 2: Function to reverse the string recursively

- This function takes three parameters: the string, **str**, the starting index, **start**, and the ending index, **end**.
- It uses a base case where it checks if **start** is less than **end**.
- If true, it swaps the characters at **start** and **end** indices and recursively calls the function with **start + 1** and **end - 1**.
- This process continues until the base case is reached, effectively reversing the string.
- **Time Complexity: O(n), where n is the length of the string. The function makes n/2 recursive calls.**
- **Space Complexity: O(n), as each recursive call adds a new frame to the call stack.**

Approach 3: Function to reverse the string using iterators

- This function uses the **reverse()** function from the **<algorithm>** library.

- It directly calls **reverse()** on the string, which reverses the characters in-place using iterators.
- **Time Complexity: O(n), where n is the length of the string. The reverse() function iterates through half of the string.**
- **Space Complexity: O(1), as the reversal is performed in-place.**

Approach 4: Function to reverse the string using reverse iterators

- This function uses reverse iterators to construct a new string by iterating from the last character to the first character of the original string.
- It creates a new string by passing the reverse iterator range to the **string()** constructor.
- **Time Complexity: O(n), where n is the length of the string. It iterates through all the characters in the string.**
- **Space Complexity: O(n), as a new string is created to store the reversed characters.**