Reverse Array [CodeStudio](https://www.codingninjas.com/studio/problems/reverse-the-array_1262298)

Given an array of integers, write a program to reverse the elements of the array in-place.

Example: Input: Array: [5, 8, 2, 11, 9]

Output: Reversed Array: [9, 11, 2, 8, 5]

Explanation: The given array [5, 8, 2, 11, 9] is reversed to [9, 11, 2, 8, 5]. The order of the elements is reversed, and the reversed array is returned.

**Approach 1: Reverses the array iteratively**

The **reverseArrayIterative** function takes a vector **arr**, a starting index **start**, and the size of the array **size** as parameters. It uses two pointers, **start** and **end**, to iterate over the array from both ends and swaps the elements until the pointers meet in the middle. This function reverses the array in-place.

**Time Complexity: The time complexity of the reverseArrayIterative function is O(N), where N is the size of the array.** It iterates over the array once, performing constant time operations (swapping elements) in each iteration.

**Space Complexity: The space complexity of reverseArrayIterative is O(1)** because they do not use any additional space that grows with the input size. They modify the original array in-place.

**Approach 2: Reverses the array recursively**

The **reverseArrayRecursive** function takes a vector **arr**, a starting index **start**, and an ending index **end** as parameters. It uses recursion to reverse the array. The base case is when **start** becomes greater than or equal to **end**. Inside the recursive call, it swaps the elements at the start and end indices and recursively calls itself with the updated indices.

**Time Complexity: The time complexity of the reverseArrayRecursive function is also O(N), where N is the size of the array.** Although the function is recursive, it performs constant time operations (swapping elements) in each recursive call, and the number of recursive calls is proportional to the size of the array.

**Space Complexity: The space complexity of reverseArrayRecursive is O(1**) because they do not use any additional space that grows with the input size. They modify the original array in-place.