Check If Array is Sorted

The given C++ program checks whether an array of integers is sorted in ascending order or not using a recursive approach. The program prompts the user to enter the size of the array and then the array elements. It then prints the array elements and determines whether the array is sorted or not based on the result of the **isSortedArray** function.

Recursive function to check if the array is sorted in ascending order

The **isSortedArray** function follows these rules:

- If the array is empty or contains only one element, it is considered sorted, and the function returns **true** (Base Case 1).
- If the first element of the array is greater than the second element, the array is not sorted, and the function returns **false** (Base Case 2).
- For larger arrays, the function calls itself recursively with a pointer to the next element and a reduced size, effectively checking if the rest of the array (excluding the first element) is sorted.

Time Complexity: The time complexity of the recursive function isSortedArray depends on the number of recursive calls made. In the worst case, it makes O(n) recursive calls, where n is the number of elements in the array. Each recursive call involves constant time operations for comparisons and pointer arithmetic. Therefore, the overall time complexity of the **isSortedArray** function is **O(n)**.

Space Complexity: The space complexity is determined by the maximum depth of the call stack during the recursive calls. In the worst case, **the call stack depth will be equal to the number of elements in the array since the function makes one recursive call per element. Therefore, the space complexity of the isSortedArray function is O(n).**

The Recursive call stack of the function