145.Write code for Insertion Sort that manages arrays with duplicate elements during the sorting process. Ensure the algorithm's behavior when encountering duplicate values, including whether it preserves the relative order of duplicates and how it affects the overall sorting outcome.

Examples:

1. **Array with Duplicates**:
   - **Input**: [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]
   - **Output**: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
2. **All Identical Elements**:
   - **Input**: [5, 5, 5, 5, 5]
   - **Output**: [5, 5, 5, 5, 5]
3. **Mixed Duplicates**:
   - **Input**: [2, 3, 1, 3, 2, 1, 1, 3]
   - **Output**: [1, 1, 1, 2, 2, 3, 3, 3]

AIM: To sort an elements by using Insertion Sort

PROGRAM:

```
def insertion_sort_with_duplicates(nums):
    n = len(nums)

    for i in range(1, n):
        key = nums[i]
        j = i - 1
        while j >= 0 and nums[j] > key:
            nums[j + 1] = nums[j]
            j -= 1

        nums[j + 1] = key
if __name__ == "__main__":
    arr1 = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]
    insertion_sort_with_duplicates(arr1)
    print("Sorted array for Test Case 1:", arr1)

    arr2 = [5, 5, 5, 5, 5]
    insertion_sort_with_duplicates(arr2)
    print("Sorted array for Test Case 2:", arr2)

    arr3 = [2, 3, 1, 3, 2, 1, 1, 3]
    insertion_sort_with_duplicates(arr3)
    print("Sorted array for Test Case 3:", arr3)
```

OUTPUT:

```
Sorted array for Test Case 1: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
Sorted array for Test Case 2: [5, 5, 5, 5, 5]
Sorted array for Test Case 3: [1, 1, 1, 2, 2, 3, 3, 3]
```

TIME COMPLEXITY: O(n^2)