159. Implement the Merge Sort algorithm in a programming language of your choice and test it on the array 12,4,78,23,45,67,89,1. Modify your implementation to count the number of comparisons made during the sorting process. Print this count along with the sorted array.

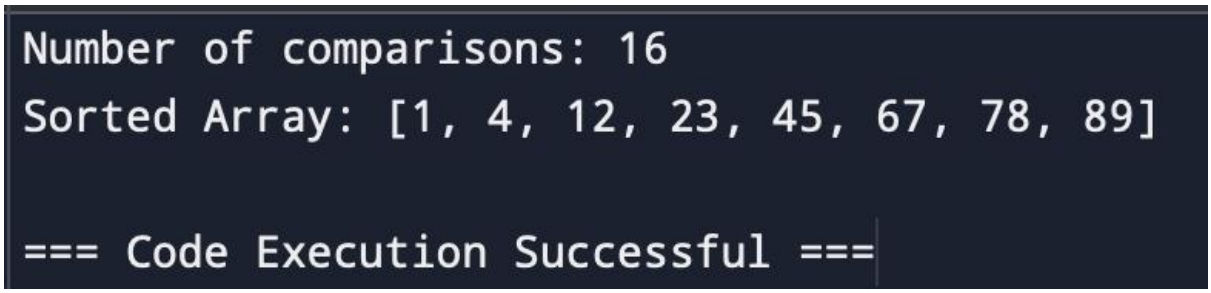
Test Cases :
Input : N= 8, a[] = {12,4,78,23,45,67,89,1}
Output : 1,4,12,23,45,67,78,89
Test Cases :
Input : N= 7, a[] = {38,27,43,3,9,82,10}
Output : 3,9,10,27,38,43,82,

PROGRAM :-

```
def merge_sort(arr):
    comparisons = 0

    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        comparisons += merge_sort(left_half)
        comparisons += merge_sort(right_half)

        i = j = k = 0

        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1
            comparisons += 1

        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

    return comparisons

def merge_sort_with_count(arr):
    comparisons = merge_sort(arr)
    return comparisons, arr

# Test the Merge Sort with Comparison Count
arr = [12, 4, 78, 23, 45, 67, 89, 1]
```

```
comparisons, sorted_arr = merge_sort_with_count(arr.copy())
print(f"Number of comparisons: {comparisons}")
print(f"Sorted Array: {sorted_arr}")
```

OUTPUT:-



TIME COMPLEXITY:- O(N log N)