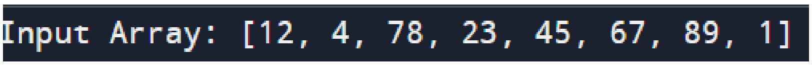
159) Implement the Merge Sort algorithm in a programming language of your choice and test it on the array 12,4,78,23,45,67,89,1. Modify your implementation to count the number of comparisons made during the sorting process. Print this count along with the sorted array.

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,

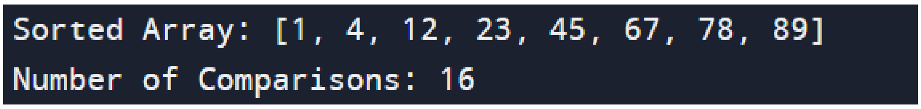
AIM : Write a program for Print this count along with the sorted array.

Program :

```
def merge_sort(arr):

    if len(arr) > 1:

        mid = len(arr) // 2

        L = arr[:mid]

        R = arr[mid:]



        # Sorting the first half

        left_comparisons = merge_sort(L)



        # Sorting the second half

        right_comparisons = merge_sort(R)



        i = j = k = 0

        comparisons = left_comparisons + right_comparisons



        while i < len(L) and j < len(R):

            comparisons += 1

            if L[i] < R[j]:

                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

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

        return comparisons
    else:
        return 0

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

# Test Cases
test_arrays = [
    [12, 4, 78, 23, 45, 67, 89, 1],
    [38, 27, 43, 3, 9, 82, 10]
]

for test_array in test_arrays:
    sorted_array, comparisons = merge_sort_with_comparisons(test_array.copy())
    print(f"Input Array: {test_array}")
    print(f"Sorted Array: {sorted_array}")
    print(f"Number of Comparisons: {comparisons}\n")
```

input : 

```
Input Array: [12, 4, 78, 23, 45, 67, 89, 1]
```

OUTPUT: 

```
Sorted Array: [1, 4, 12, 23, 45, 67, 78, 89]
Number of Comparisons: 16
```

TIME COMPLEXITY: O(n logn)