Worksheet 1.3

Student Name: Ayush Yadav
UID: 25IMC13004
Branch: MCA (AI & ML)
Section/Group: MAM-1 (A)
Semester: I
Date of Performance: 09/09/25
Subject Name: Design and Analysis of Algorithms Lab
Subject Code: 25CAP-612

1. Aim/Overview of the practical:

Experiment-1:
Minimum Cost Spanning Tree of an Undirected Graph.

2. Coding :

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2

    left_half = merge_sort(arr[:mid])
    right_half = merge_sort(arr[mid:])

    return merge(left_half, right_half)
```

```
def merge(left, right):
    sorted_arr = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            sorted_arr.append(left[i])
            i += 1
        else:
            sorted_arr.append(right[j])
            j += 1


    sorted_arr.extend(left[i:])
    sorted_arr.extend(right[j:])

    return sorted_arr

arr = []
num = int(input("Enter number of elements of array:"))
for i in range(num):
    element= int(input(f"Enter element {i+1}:"))
    arr.append(element)
print("Original array:", arr)
print("Sorted array:", merge_sort(arr))
```

OUTPUT:-

```
PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/DAA/Merge Sort.py"
Enter number of elements of array:5
Enter element 1:8
Enter element 2:3
Enter element 3:5
Enter element 4:1
Enter element 5:6
Original array: [8, 3, 5, 1, 6]
Sorted array: [1, 3, 5, 6, 8]
PS E:\CLASS CODING>
```

3. Learning outcomes (What I have learnt):

- Understand the working principle of the Merge Sort algorithm.
- Learn how to implement Merge Sort using divide and conquer strategy.
- Analyze the time and space complexity of Merge Sort.
- Compare Merge Sort with other sorting algorithms in terms of efficiency.
- Develop problem-solving skills using divide-and-conquer techniques.