**Practical -4**

**Aim:** Write program to sort given data set using Merge sort or Quicksort to demonstrate

Divide and Conquer techniques

**Theory:**

**Merge Sort:**

Merge Sort is a sorting algorithm that follows the Divide and Conquer paradigm. It divides the input array into smaller subarrays, sorts these subarrays recursively, and then merges them to produce a sorted array. The key steps involve dividing the array, sorting the divided parts, and merging the sorted parts to achieve the final sorted array.

The algorithm's efficiency comes from its ability to break the sorting process into smaller, more manageable parts, and then combine them efficiently by merging the sorted subarrays

**Acceptance Criteria:**

The acceptance criteria for this code involve verifying the successful implementation of Merge Sort with a class-based interface and user-input data:

* The program should prompt the user to input a list of elements.
* The Merge Sort algorithm should be implemented correctly within a class structure.
* The input data should be sorted in ascending order.
* The program should display the sorted list of elements.

**Code:**

class MergeSort:

def merge(self, arr, left, mid, right):

n1 = mid - left + 1

n2 = right - mid

L = [0] \* n1

R = [0] \* n2

for i in range(n1):

L[i] = arr[left + i]

for j in range(n2):

R[j] = arr[mid + 1 + j]

i = 0

j = 0

k = left

while i < n1 and j < n2:

if L[i] <= R[j]:

arr[k] = L[i]

i += 1

else:

arr[k] = R[j]

j += 1

k += 1

arr[k] = L[i]

i += 1

k += 1

while j < n2:

arr[k] = R[j]

j += 1

k += 1

def merge\_sort(self, arr, left, right):

if left < right

mid = (left + (right - 1)) // 2

self.merge\_sort(arr, left, mid)

self.merge\_sort(arr, mid + 1, right)

self.merge(arr, left, mid, right)

def sort(self, arr):

self.merge\_sort(arr, 0, len(arr) - 1)

def input\_list ():

print ("Enter elements separated by space:")

elements = list (map (int, input (). split ()))

return elements

Print ("Enter the data to be sorted:")

data = input\_list ()

merge\_sort = MergeSort ()

merge\_sort.sort(data)

print ("Sorted data using Merge Sort:")

print(\*data)

**Input:**

Enter the data to be sorted:

Enter elements separated by space:

7 8 1 4 26 55 82

**Output:**

Sorted data using Merge Sorted : 1 4 7 8 26 55 82

**Conclusion:**

The provided Python code demonstrates the Merge Sort algorithm through a class-based implementation. It effectively takes user input for a list of elements, sorts them using the Merge Sort technique utilizing the Divide and Conquer strategy, and displays the sorted list as the output. To validate the acceptance criteria, one can run the program with various input lists, ensuring that it sorts the elements correctly using the Merge Sort algorithm and displays the expected output of a sorted list. The code showcases the efficiency of the Divide and Conquer approach in sorting large datasets by breaking down the sorting process into smaller, manageable parts.