**Data Structures and Algorithms**

**Lab 9**

**Submitted To:**

Mr. Dilshad Sabir

**Submitted By:**

Manaal Waseem

FA18-BCE-074

**In Lab:**

**Task 1:**

***Complete the merge() function for Merge Sort*.**

**Program:** In this program, ***“void merge(int \* ptr\_arrA, int sizeA, int \* ptr\_arrB, int sizeB)”*** has been completed by implementing a for loop that copies the sorted elements in array C to original array A.



**Output:**



**Task 2:**

***Complete the partition() function for Quick Sort.***

**Program:** In this program, function ***“int partition(int \* ptr\_array, int start\_idx, int end\_idx)”*** takes last element as pivot, places the pivot element at its correct position in sorted array, and places all smaller (smaller than pivot) elements to left of pivot and all greater elements to right of pivot.

****

**Output:**



**Post Lab:**

***Study and perform comparative analysis between different sorting algorithms we have implemented in current and previous Lab.***

* **Bubble sort and Insertion sort –**  
  Average and worst case time complexity: n^2  
  Best case time complexity: n when array is already sorted.  
  Worst case: when the array is reverse sorted.
* **Selection sort –**  
  Best, average and worst case time complexity: n^2 which is independent of distribution of data.
* **Merge sort –**  
  Best, average and worst case time complexity: nlogn which is independent of distribution of data.
* **Quick sort –**  
  It is a divide and conquer approach with recurrence relation:

T(n) = T(k) + T(n-k-1) + cn

Worst case: when the array is sorted or reverse sorted, the partition algorithm divides the array in two subarrays with 0 and n-1 elements. Therefore,

T(n) = T(0) + T(n-1) + cn

Solving this we get, T(n) = O(n^2)

Best case and Average case: On an average, the partition algorithm divides the array in two subarrays with equal size. Therefore,

T(n) = 2T(n/2) + cn

Solving this we get, T(n) = O(nlogn)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THE END**