a. Explanation of status and stopping point, if incomplete.

We were successfully able to complete the implementation of all the required algorithms.
The algorithms that were implemented as a part of this project are:-

1) Bubble Sort - $O(n^2)$
2) Merge Sort - O(n log(n))
3) Counting Sort - O(n+k)
4) Longest Common Subsequence - O(n^3)

We also implemented a Random Number Generator. This program produces a sequence of random numbers

b. Explanation of additional functions and analysis, if any.

Analysis:
Firstly taking each case (best, average and worst) into account:

1) **Bubble Sort:**

We observed that experimental running time is of the same order as predicted by the asymptotic analysis. As depicted in the graphs in the previous question, we can see that running time of the algorithm for different cases is as follows:

BEST CASE > AVERAGE CASE > WORST CASE

which is expected.

Algorithm	Time Complexity:Best	Time Complexity:Average	Time Complexity:Worst
Bubble sort	O(n)	$O(n^2)$	$O(n^2)$

2) **Merge Sort:**

We observed that experimental running time is of the same order as predicted by the asymptotic analysis. As depicted in the graphs in the previous question, we can see that running time of the algorithm for different cases is as follows:

BEST CASE > AVERAGE CASE > WORST CASE

which is expected.

Algorithm	Time Complexity:Best	Time Complexity:Average	Time Complexity:Worst
Merge sort	O(n log(n))	O(n log(n))	O(n log(n))

3) **Counting Sort:**

We observed that experimental running time is of the same order as predicted by the asymptotic analysis. As depicted in the graphs in the previous question, we can see that running time of the algorithm for different cases is as follows:

BEST CASE > AVERAGE CASE > WORST CASE

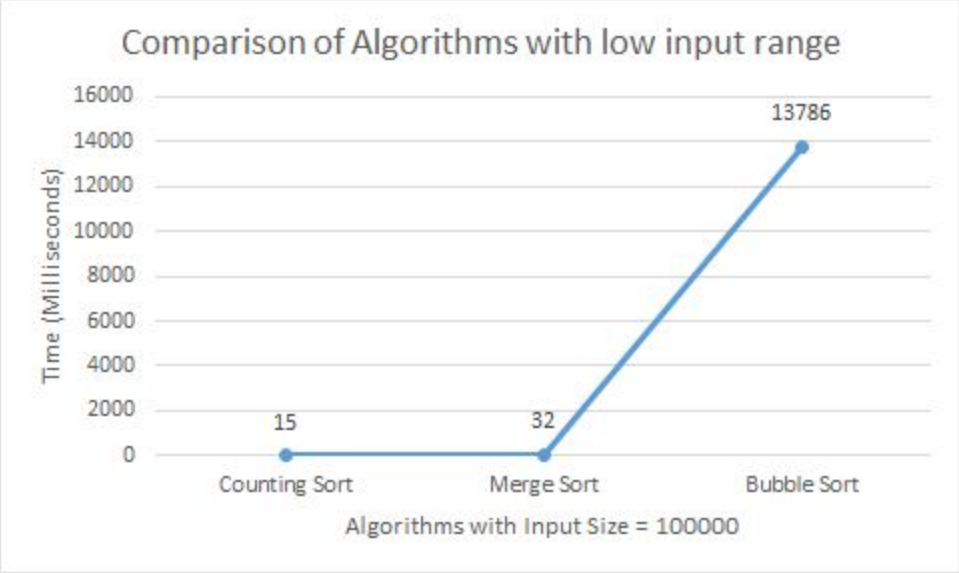
which is expected.

Algorithm	Time Complexity:Best	Time Complexity:Average	Time Complexity:Worst
Counting sort	O(n+k)	O(n+k)	O(n+k)

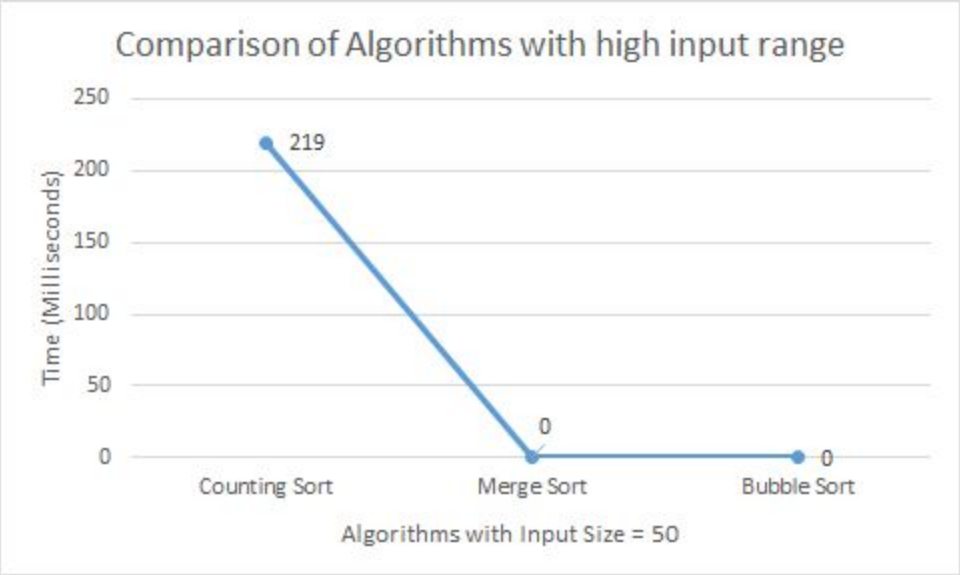
Now comparing algorithms with each other, we draw conclusion that Counting Sort consumes less time than Bubble and Merge Sort. This can be explained from the readings and graphs in the previous question. When we tried to run the algorithms with different input sizes and different types of files i.e. sorted input file, reverse sorted input file and random input file we concluded that the running time of algorithms is as follows:

COUNTING SORT > MERGE SORT > BUBBLE SORT

Generalised graph for the comparison:



But there is a special scenario where Counting Sort is not efficient as compared to Merge Sort. Counting sort is fast because of the way that elements are sorted using their values as array keys. This means that more memory is required for the extra array at the cost of running time. It runs in O(n + k) time where n is the number of elements to be sorted and k is the number of possible values in the range. So, during running the algorithm for this scenario with the input file with higher range, we concluded that Merge Sort is more efficient than Counting Sort.



c. Contribution statement detailing the contribution of each group member in completing this project.

We (Joyta and Vipul) distributed the task in such a manner that each one of us contributes equally. The implementation of all the algorithms was simultaneous work of each one of us. We held meeting and discussed every aspect required for the project. The final project report is also combined contribution of both of us.