Practice Assignment

Module 4	
Sr. No.	**Questions**
1	Explain the design principles and strategies behind various sorting algorithms, such as bubble sort, selection sort, insertion sort, merge sort, and quicksort. Discuss the time and space complexities of each algorithm and their suitability for different types of data.
2	In the context of sorting algorithms, describe the differences between in-memory sorting and sequential storage sorting. How do these approaches impact the performance and scalability of sorting algorithms? Provide examples of scenarios where each approach would be preferred.
3	Consider the scenario of sorting short lists versus sorting long lists. How does the choice of sorting algorithm affect the efficiency of sorting in these two cases? Discuss the trade-offs between different sorting algorithms when dealing with lists of varying lengths.
4	Explore the concept of semi-sorted lists and their relevance in sorting algorithms. How do sorting algorithms perform when applied to semi-sorted lists compared to completely random lists? Provide examples of real-world situations where semi-sorted lists are commonly encountered.

5	Dive into the world of sorting algorithms used in databases. Discuss the challenges and considerations when sorting large datasets that cannot fit entirely in memory. How do external sorting algorithms address these challenges, and what are their advantages over traditional in-memory sorting algorithms?
6	Analyse the strengths and weaknesses of the bubble sort algorithm. How does its performance compare to other sorting algorithms, and in what situations might it be a suitable choice despite its simplicity? Provide examples where bubble sort can outperform more complex algorithms.
7	Investigate the characteristics of the selection sort algorithm. How does it compare to other sorting algorithms in terms of its efficiency and ease of implementation? Discuss the impact of the input data's initial order on the performance of selection sort.
8	Examine the insertion sort algorithm and its behaviour on partially sorted data. How does its performance differ from other sorting algorithms, especially when dealing with data that is nearly sorted? Illustrate this with real-world scenarios where insertion sort shines.
9	Consider the scenario of sorting data with duplicate values using different sorting algorithms. How do algorithms like quicksort and merge sort handle duplicates, and how does this impact their performance and stability? Provide insights into the trade-offs involved.
10	Explore the divide-and-conquer approach employed by the merge sort algorithm. Explain how this algorithm works step by step, including the process of merging sorted sublists. Discuss

	its time and space complexity and situations where it performs well or struggles.