Prepare a Lab sheet of all practical work from:

Chapter1

1. A program to read n numbers and to find the sum and average of those numbers.

Chapter 2

1. WAP to demonstrate Stack
2. WAP to Demonstrate infix to postfix expression. (User Have to insert data manually.)
3. WAP to evaluate prefix expression.

Chapter 3

1. WAP to demonstrate queue. Users must insert the Max size of the queue and while enqueue and dequeue users must provide data manually. (not hard code). Explain the algorithm of dequeue and enqueue.
2. WAP to demonstrate Circular queue.
3. Demonstrate how the priority queue helps to deque elements based on their priority.

Chapter 4

1. Write a program to calculate the factorial of a given number using recursion.
2. A program to find the n-th term of the Fibonacci sequence using recursion.
3. A program to compute the Greatest Common Divisor (GCD) of two numbers using recursion.
4. A program to find the largest element in an array using tail recursion.
5. A program to solve the Tower of Hanoi problem using recursion.

Chapter 5

1. A program to implement a basic list using an array and perform insertions and deletions at the beginning, end, and a specified position.
2. A program to implement a singly linked list and perform operations such as node creation, insertion, and deletion from the beginning, end, and a specified position.
3. A program to implement a doubly linked list and demonstrate insertion and deletion at both ends and in the middle.
4. A program to implement a circular linked list and perform operations like insertion at the beginning, end, and a specified position.
5. A program to implement a stack and a queue using linked lists and demonstrate basic operations such as push, pop (for stack), and enqueue, dequeue (for queue).

Chapter 6

1. Given a list of integers, implement internal sorting using Bubble Sort and display the sorted list.
2. A program to implement Selection Sort on a list of integers and print the sorted list.
3. Implement Insertion Sort to sort a list of customer IDs in ascending order and display the sorted IDs.
4. Sort a list of student scores using Shell Sort and compare its execution time with Bubble Sort for the same list.
5. Implement Merge Sort to sort a large array of random integers and display the sorted array. Measure and display the time taken for the sorting process.
   Sort an array of random integers using Quick Sort and compare the time complexity with Merge Sort by testing both algorithms on arrays of different sizes.
6. Sort an array of integers using Heap Sort and test its performance with datasets of different sizes. Compare the results with Quick Sort and Merge Sort.
7. Write a program to implement the following sorting algorithms and sort a given list of integers:
   Bubble Sort
- Selection Sort
- Insertion Sort
8. Implement Shell Sort and test it on a set of random numbers.

Chapter 7

1. Write a program to implement
- Sequential Search
- Binary Search

Compare the efficiency of Sequential and Binary Search for sorted and unsorted arrays.
Compare their execution times for a large dataset. Implement collision resolution techniques:

- Linear Probing
- Quadratic Probing
- Chaining (using linked lists)

Chapter 8

1. Write a program to implement Depth-first Search in C programming with algorithms.
2. Write a program to implement Breadth-first Search in C programming with algorithms