**Data Structures**

* **Linked List:**

1. Perform following operations on a singly linked list:

* Insert a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Delete a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Display the List.

1. Perform following operations on a singly circular linked list:

* Insert a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Delete a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Display the List.

1. Perform following operations on a doubly linked list:

* Insert a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Delete a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Display the List.

4. Perform following operations on a doubly circular linked list:

* Insert a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Delete a Node:

1. At Beginning of List.
2. At End of List.
3. After a key value.

* Display the List.

**Exercises:**

1. Program to create an ordered linked list.
2. Program to perform intersection of two sets of lists.
3. Program to perform union of two sets of lists.
4. Program to reverse a linked list.
5. Program to check whether a linked list is a palindrome or not.
6. Program to perform addition of two polynomials.
7. Program to perform multiplication of two polynomials.
8. Program to remove duplicate values from a linked list.(Use Doubly Linked List)
9. Program to sort a given linked list.(Use Doubly Circular Linked List)
10. Program to split a given linked list(containing numbers),into two lists of odd and even numbers respectively.(Use Singly Circular Linked List)

* **Stack:**

1. Perform following operations on a stack using Array:

* Push
* Pop

2. Perform following operations on a stack using Linked List:

* Push
* Pop

**Exercises:**

1. Check whether a symbol is balanced or not using stack.

2. Convert an infix expression into postfix expression using stack.

* **Queue**:

1. Perform following operations on simple queue(use Array):
2. Insert
3. Delete

2. Perform following operations on a circular queue(use Array):

1. Insert

B. Delete

3. Perform following operations on simple queue(use Linked List):

A. Insert

B. Delete

4. Perform following operations on a circular queue(use Linked List):

1. Insert

B. Delete

* **Tree:**

1. Write a program to perform the following:
2. Create a Binary Search tree.
3. Traverse through the created tree using following methods(both iterative and recursive):

* Pre-Order Traversal
* In-Order Traversal
* Post-Order Traversal

1. Find minimum and maximum value from the binary search tree.
2. Delete a node from the tree.

2. Write a program to create an expression tree and perform traversals to display expression..

* **Algorithms:**

1. Write a program to sort a list of numbers using Bubble Sort.
2. Write a program to sort a list of numbers using Selection Sort.
3. Write a program to sort a list of numbers using Insertion Sort.
4. Write a program to search for a value from a list using Linear Search.
5. Write a program to search for a value from a list using Binary Search.
6. Write a program to sort a list of numbers using Radix Sort.
7. Write a program to sort a list of numbers using Heap Sort.
8. Write a program to sort a list of numbers using Shell Sort.
9. Write a program to sort a list of numbers using Quick Sort.
10. Write a program to sort a list of numbers using Merge Sort.

* **Graph:**

1. Write a program to find the shortest path between two vertices in an unweighted directed graph.
2. Write a program to find the shortest path between two vertices in a weighted directed graph.
3. Write a program to find the minimum spanning tree of a graph using Prim’s Algorithm.
4. Write a program to find the minimum spanning tree of a graph using Kruskal’s Algorithm.
5. Write a program to perform Breadth-First Traversal on a graph.
6. Write a program to perform Depth-First Traversal on a graph.