Data Structure C++

1. Linked List Overview: Intro To Linked List

Problem Statement: Linked List Creation

Write a program to create a basic linked list that contains integers. Define a structure for each node, and implement functions to create, display, and manage the linked list.

2. Appending To A Linked List
Problem Statement: Appending Nodes

Extend the previous program by implementing a function that appends a new node with a given value to the end of the linked list.

3. Prepending To A Linked List

Problem Statement: Prepending Nodes

Enhance the program to include a function that prepends a new node with a given value to the beginning of the linked list.

4. Printing Out Our Linked List

Problem Statement: Display Linked List

Modify the program to add a function that prints out the elements of the linked list.

5. Remove First Node From Our Linked List

Problem Statement: Removing First Node

Implement a function to remove the first node from the linked list.

6. Remove Last Node From Our Linked List

Problem Statement: Removing Last Node

Add a function that removes the last node from the linked list.

7 Remove Node At A Certain Position From Our Linked List

Problem Statement: Removing Node at Position

Develop a function that removes a node at a specified position in the linked list.

8. Insert Node At A Certain Position Within Our Linked List

Problem Statement: Inserting Node at Position

Implement a function to insert a new node with a given value at a specified position in the linked list.

9. Linked List Destructor

Problem Statement: Linked List Cleanup

Extend the linked list program to include a destructor that releases the memory allocated for the linked list nodes.

10. Doubly Linked List Overview: Intro to Doubly Linked List

Problem Statement: Doubly Linked List Creation

Write a program to create a basic doubly linked list that contains integers. Define a structure for each node, and implement functions to create, display, and manage the doubly linked list.

11. Prepending To A Doubly Linked List

Prob13 and 18 lem Statement: Prepending Nodes

Enhance the previous program by implementing a function that prepends a new node with a given value to the beginning of the doubly linked list.

12.13. Insert Node At A Certa Appending To A Doubly Linked List

Problem Statement: Appending Nodes

Extend the program to include a function that appends a new node with a given value to the end of the doubly linked list.
Modify the program to add a function that prints out the elements of the doubly linked list in reverse order.

in Position Within Our Doubly Linked List

Problem Statement: Inserting Node at Position

Implement a function to insert a new node with a given value at a specified position in the doubly linked list.

14. Printing Out Our Doubly Linked List In Reverse

Problem Statement: Displaying in Reverse

15. Remove First Node From Our Doubly Linked List

Problem Statement: Removing First Node

Implement a function to remove the first node from the doubly linked list.

16. Remove Node At A Certain Position From Our Doubly Linked List

Problem Statement: Removing Node at Position

Develop a function that removes a node at a specified position in the doubly linked list.

17. Remove Last Node From Our Doubly Linked List

Problem Statement: Removing Last Node

Add a function that removes the last node from the doubly linked list.

18. Insert Node At A Certain Position Within Our Doubly Linked List

Problem Statement: Inserting Node at Position

Implement a function to insert a new node with a given value at a specified position in the doubly linked list.

19. Stack Overview

Problem Statement: Understanding Stack Operations

Write a program that provides a menu-based interface for performing stack operations: push, pop, and display the elements. Use an array to implement the stack data structure.

20. Implementing a Stack using an Array

Problem Statement: Array-based Stack Implementation

Create a program that implements a stack using an array. Implement push and pop operations along with displaying the stack's contents. Test the implementation by performing various stack operations.

21. Implementing a Stack using a Linked List

Problem Statement: Linked List-based Stack Implementation

Extend the previous program by implementing a stack using a linked list. Implement push and pop operations along with displaying the stack's contents. Test the implementation by performing various stack operations

22. Implementing a Queue using an Array

Problem Statement: Array-based Queue Implementation

Write a program to implement a queue using an array. Implement enqueue, dequeue, and display operations for the queue. Test the implementation by performing various queue operations.

23. Implementing a Queue using a Linked List

Problem Statement: Array-based Queue Implementation

Create a program that implements a queue using a linked list. Implement enqueue, dequeue, and display operations for the queue. Test the implementation by performing various queue operations.

24. Binary Search Tree Overview Problem Statement: Understanding Binary Search Trees Write a program that explains the concept of binary search trees and demonstrates how they maintain the properties of the left subtree containing values less than the root, and the right subtree containing values greater than the root.

25. Binary Search Tree Insert Overview Problem Statement: Insertion in Binary Search Tree Develop a program that provides an overview of inserting nodes into a binary search tree. Explain the process of maintaining the binary search tree property while inserting nodes.

26. Binary Search Tree Insert Method Problem Statement: Binary Search Tree Insertion Write a program that implements a binary search tree and defines a method to insert nodes into the tree while maintaining the binary search tree property. Display the tree after inserting nodes.

27. Binary Search Tree Deletion Overview Problem Statement: Deletion in Binary Search Tree Create a program that explains the process of deleting nodes from a binary search tree while maintaining the binary search tree property.

28. Binary Search Tree Deletion Method Problem Statement: Binary Search Tree Deletion Develop a program that implements a binary search tree and defines a method to delete nodes from the tree while maintaining the binary search tree property. Display the tree after deletion operations.

29. Binary Min/Max Heap Overview Problem Statement: Understanding Binary Min/Max Heaps Write a program that explains the concept of binary min/max heaps and demonstrates how they maintain the property of parent nodes having values smaller/larger than their child nodes.

30. Binary Min/Max Heap Insert Overview Problem Statement: Insertion in Binary Min/Max Heap Develop a program that provides an overview of inserting elements into a binary min/max heap. Explain the process of maintaining the min/max heap property while inserting elements.

31. Binary Min/Max Heap Insert (Array Recursive Implementation) Problem Statement: Binary Min/Max Heap Insertion Write a program that implements a binary min/max heap using an array and defines a recursive method to insert elements into the heap while maintaining the min/max heap property. Display the heap after inserting elements.

32. Binary Min/Max Heap Deletion Overview Problem Statement: Deletion in Binary Min/Max Heap Create a program that explains the process of deleting elements from a binary min/max heap while maintaining the min/max heap property.

33. Binary Min/Max Heap Deletion (Array Recursive Implementation) Problem Statement: Binary Min/Max Heap Deletion Develop a program that implements a binary min/max heap using an array and defines a recursive method to delete elements from the heap while maintaining the min/max heap property. Display the heap after deletion operations