**Question 1: Theory Questions (30 Marks)**

1. **Copy Semantics vs Move Semantics (6 Marks)**  
   a. Define **copy semantics** and **move semantics**.  
   b. Explain the key differences between copy semantics and move semantics with respect to memory management.  
   c. Provide example code that demonstrates **copy semantics** and **move semantics** in C++.
2. **Deep Copy vs Shallow Copy (6 Marks)**  
   a. What is the difference between **deep copy** and **shallow copy**?  
   b. Discuss the advantages and disadvantages of each.  
   c. Provide an example where **deep copy** is necessary and another example where **shallow copy** is sufficient.
3. **R-Value and L-Value (6 Marks)**  
   a. Explain the terms **L-value** and **R-value** in C++.  
   b. How do R-values and L-values relate to **move semantics**? Provide an example.  
   c. Discuss the impact of L-values and R-values on **memory handling** and **efficiency**.
4. **Const Qualifier (6 Marks)**  
   a. What does the **const** qualifier mean in C++?  
   b. Explain its effects when used with functions, variables, and pointers. Provide examples of each.  
   c. What is the role of **constant member functions** in classes? Explain with an example.
5. **Abstraction (6 Marks)**  
   a. Define **abstraction** in the context of object-oriented programming.  
   b. Discuss how **copy constructors**, **move constructors**, and **assignment operators** play a role in abstraction in C++.  
   c. Explain how the **Rule of Five** applies to class design with respect to resource management.

**Question 2: Linked List (60 Marks)**

1. **Linked List Implementations (30 Marks)**  
   a. **Singly Linked List** (10 Marks)  
   Implement a **singly linked list** that supports the following operations:
   * **Insertion** at the front and back.
   * **Deletion** from the front and back.
   * **Display** the list.
   * Provide an iterator for traversal.

b. **Doubly Linked List** (10 Marks)  
Explain how a **doubly linked list** differs from a singly linked list.  
Implement a **doubly linked list** that supports:

* + **Insertion** and **deletion** operations.
  + **Traversal** using an iterator (both forward and backward).

c. **Circular Linked List** (10 Marks)  
Explain what a **circular linked list** is.  
Implement a **circular singly linked list** that supports:

* + **Insertion** at the front and back.
  + **Traversal** (ensuring it loops correctly).

1. **Iterator Implementations (30 Marks)**  
   Implement iterators for both **singly** and **doubly linked lists**:

a. **Forward Iterator** (15 Marks)

* + Implement a **forward iterator** for the singly linked list.
  + The iterator should support:
    - **Dereference (\*)**
    - **Pre-increment (++ before value)**
    - **Post-increment (++ after value)**
    - **Equality (==)** and **inequality (!=)**
    - **begin()** and **end()** methods.

b. **Bidirectional Iterator** (15 Marks)

* + Implement a **bidirectional iterator** for the doubly linked list.
  + The iterator should support:
    - **Dereference (\*)**
    - **Pre-increment (++) and Post-increment (++)**
    - **Pre-decrement (--) and Post-decrement (--)**
    - **Equality (==)** and **inequality (!=)**
    - **begin()** and **end()** methods.

**Question 3: Tree (60 Marks)**

1. **Tree Types and Implementations (30 Marks)**  
   a. **Binary Tree** (10 Marks)
   * Define a **binary tree** and implement the following:
     + **Insertion** of nodes.
     + **Pre-order**, **In-order**, and **Post-order** traversal methods.
     + **Search** for a specific value in the tree.

b. **N-ary Tree** (10 Marks)

* + Define what an **N-ary tree** is.
  + Implement an **N-ary tree** where each node can have multiple children.
  + Provide a function for **adding a child** to a specific node.
  + Implement **level-order traversal** for displaying the tree.

c. **B-Tree** (10 Marks)

* + Explain what a **B-tree** is and how it differs from binary trees.
  + Implement the **insertion** operation for a **B-tree** with a minimum degree of 2.
  + Implement **level-order traversal** for the B-tree.

1. **Traversal Techniques (15 Marks)**  
   a. **Pre-order Traversal** (5 Marks)
   * Explain and implement **pre-order traversal** for binary trees and N-ary trees.

b. **In-order Traversal** (5 Marks)

* + Explain and implement **in-order traversal** for binary trees (applicable only for binary search trees).

c. **Post-order Traversal** (5 Marks)

* + Explain and implement **post-order traversal** for binary trees and N-ary trees.

1. **Tree Attributes (15 Marks)**  
   a. Define and explain the following terms in the context of trees:
   * **Height**
   * **Depth**
   * **Degree**
   * **Leaf nodes**
   * **Child nodes**

b. Write a function to compute:

* + The **height** of a binary tree.
  + The **depth** of a node in a binary tree.

**General Topics to Study**

1. **Pointers**:
   * Understand how **pointers** work in linked lists and tree structures.
   * **Memory management** and **dereferencing**.
2. **Indexers**:
   * Study the use of **indexers** (used for accessing nodes in data structures like linked lists).
3. **Recursion**:
   * Theory and application of **recursive functions** for tree traversal.
4. **Big-O Notation**:
   * Be able to analyze the performance of common algorithms for trees, linked lists, and other data structures.