Data Structure and Algorithms Lab

Lab Program 3

1. A music streaming application needs to maintain a playlist where users can play songs in both forward and backward directions. To support this, the application should use a doubly linked list where each node stores the song title. The structure should allow adding new songs to the playlist, removing songs, and displaying the playlist in both forward and reverse order. Since a doubly linked list maintains links in both directions, it becomes easy for the user to navigate back and forth across the songs, delete specific songs, or insert new ones at desired positions.

2. An online bookstore wants to manage its collection of books efficiently so that searching, inserting, and deleting book IDs can be done quickly. To achieve this, the system should use a Binary Search Tree (BST) where each node stores a unique book ID. The BST will ensure that for any node, all book IDs in the left subtree are smaller and all book IDs in the right subtree are larger. This structure will allow the bookstore to maintain its inventory dynamically by adding new books, removing discontinued ones, and searching for specific book IDs in an efficient manner. Additionally, the program should support different traversal methods (inorder, preorder, and postorder) to display the books in sorted order, insertion order, or hierarchical order, depending on the requirement.

3. A university library is designing a system to maintain its digital catalog of journal IDs. Since the catalog is frequently updated with new journals being added and outdated ones being removed, the data structure must support efficient insertion and searching while keeping the catalog balanced to avoid performance degradation. To achieve this, the system should be implemented using an AVL Tree, a self-balancing binary search tree where the heights of the left and right subtrees of any node differ by at most one. The program should also allow traversal of the AVL tree (inorder, preorder, postorder) to display the catalog in sorted or hierarchical order.

Additional Questions:

1. Design and implement a menu-driven Java program to perform operations on a doubly linked list. The program should allow the user to Insert a node at the beginning of the list, Insert a node at the end of the list, Insert a node at a specific position, Delete a node from the beginning, Delete a node from the end, Delete a node from a specific position, Traverse and display the list forward, Traverse and display the list backward, Check whether the list is empty. The program should handle cases such as inserting into an empty list, deleting from an empty list, and attempting to delete at invalid positions with proper error messages.

2. In airline reservation system needs to store and manage passenger booking IDs in such a way that the data remains well-organized and quickly searchable even as thousands of bookings are inserted and cancelled every day. If the system were to use a simple binary search tree, continuous insertions in sorted order could cause the structure to become skewed, leading to poor performance. To prevent this, the system must be designed so that after every insertion or deletion, the structure automatically adjusts itself to ensure that no branch becomes significantly taller than the others. This balancing guarantees that search, insertion, and deletion operations always run efficiently. Additionally, the system should support displaying the booking IDs in sorted order (inorder traversal) and also allow preorder and postorder traversals for hierarchical reporting.