**Lab Report on AVL Tree and Splay Tree Implementation**

**Objective:**

The objective of this lab is to implement and test AVL (Adelson-Velsky and Landis) Tree and Splay Tree data structures in C++. The focus is on understanding tree operations such as insertion, deletion, searching, finding minimum and maximum values, and checking for the presence of a key.

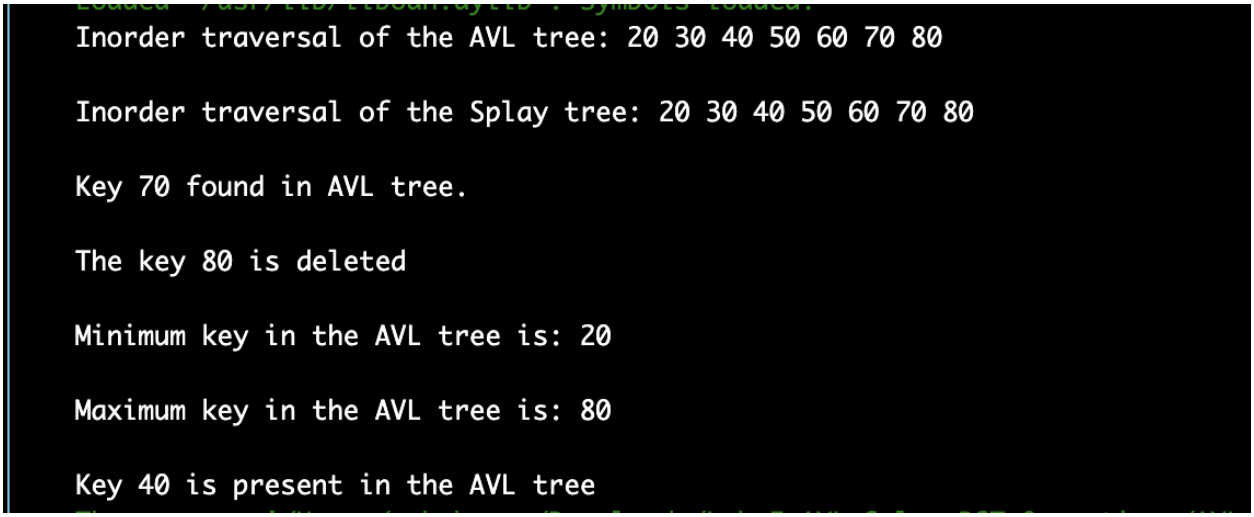
**Methodology:**

- **Implementation**:
  - Two classes were implemented: **AVLTree** and **SplayTree**.
  - **AVLTree** includes methods for insertion, in-order traversal, deletion, finding minimum and maximum values, and checking if a key is present.
  - **SplayTree** focuses on insertion and in-order traversal.
  - Both classes employ a nested **Node** structure to represent individual elements in the tree.
- **Key Methods in AVLTree**:
  - **insert()**: Inserts a new key into the tree.
  - **deleteNode()**: Removes a key from the tree.
  - **findMin()** and **findMax()**: Find the minimum and maximum key in the tree, respectively.
  - **isPresent()**: Checks if a particular key is present in the tree.
  - **inorder()**: Performs an in-order traversal of the tree.
- **Testing**:
  - The **main** function tests various operations on the AVL and Splay trees.
  - A set of keys **[50, 30, 70, 20, 40, 60, 80]** is inserted into both trees.
  - In-order traversal is performed to display the tree structure.
  - Specific operations such as searching for a key, finding minimum and maximum keys, and checking the presence of a key are executed on the AVL tree.

**Results:**

- **AVL Tree Operations**:
  - The in-order traversal successfully displayed the AVL tree in ascending order.
  - The search operation was able to find or report the absence of a key correctly.
  - Deletion of a key (80) was successfully performed and confirmed with a print statement.
  - The minimum and maximum key values were correctly identified.
  - The presence check for a specific key (40) accurately determined whether the key was in the tree.
- **Splay Tree Operations**:
  - The in-order traversal of the Splay tree was successful, displaying the tree's structure.

- **Output:**




**Discussion:**

- The AVL tree operations, including insertion, deletion, and search, maintained the tree's balanced property, as evidenced by the successful in-order traversals.
- The deletion function in the AVL tree included print statements confirming the key being deleted, enhancing the understanding of the operation.
- The Splay tree was primarily tested for insertion and traversal, showcasing its dynamic nature.
- The **isPresent** method efficiently checked for the existence of keys, demonstrating effective tree traversal.

**Conclusion:**

The lab successfully implemented and tested key operations in AVL and Splay trees. The AVL tree operations were particularly notable for maintaining balance and efficiently handling various tree manipulations. The Splay tree's behavior in response to operations was as expected. This lab reinforces the understanding of tree-based data structures and their applications in efficient data storage and retrieval.