Binary Search Tree Implementation

1. Purpose:

- The purpose of this project is to implement a Binary Search Tree (BST) in C++ using smart pointers (`std::unique_ptr`) for managing memory and ensuring ownership semantics.

2. Components:

- Node Structure: Defines the structure for a tree node (`Node`) with integer data (`int val`) and pointers to left and right children (`std::unique_ptr<Node> left, right`).
- Binary Search Tree Class (`BinarySearchTree`):
  - Constructor (`BinarySearchTree()`): Initializes an empty tree (`root` is `nullptr`).
  - Insertion (`insert(int val)`): Inserts a value into the BST maintaining the BST property using a recursive helper function (`insertRecursive`).
  - Search (`search(int val)`): Searches for a value in the BST using a recursive helper function (`searchRecursive`).
  - Deletion (`deleteNode(int val)`): Deletes a node with a given value from the BST using a recursive helper function (`deleteNodeRecursive`). Handles cases where the node to be deleted has zero, one, or two children.
  - Inorder Traversal (`inorder()`): Prints the inorder traversal of the BST using a recursive helper function (`inorderRecursive`).

3. Implementation Details:

- Insertion: Recursively places the value in the correct position based on the BST property.
- Search: Recursively searches for the value starting from the root.

- Deletion: Uses recursive deletion based on the number of children the node to be deleted has. If it has two children, it finds the inorder successor, replaces the node's value with the successor's value, and deletes the successor node recursively.
- Inorder Traversal: Prints values in ascending order.

4. Usage Example (`main` function):

- Creates an instance of `BinarySearchTree`.
- Inserts several values (`5, 2, 8, 3, 9`) into the tree.
- Prints the inorder traversal before and after deleting a value (`8`).
- Searches for a value (`8`) and confirms its presence or absence.

5. Output:

- Shows the inorder traversal of the tree before and after deletion.
- Reports whether a specific value is found in the tree.

6. Conclusion:

- The implementation demonstrates the basic operations of a Binary Search Tree using modern C++ features (`std::unique_ptr` for memory management) efficiently.
-