checks if the current `node` is `nullptr`, indicating an empty spot in the tree. If it is `nullptr`, a new node is created with the given `value`, and the `node` pointer is updated to point to this new node. If the `node` is not `nullptr`, the function recursively calls itself to traverse the tree further. If the `value` is less than the current node's data, it continues the insertion process on the left subtree; otherwise, it continues on the right subtree.

- `inOrderTraversalRecursive(Node* node) const`: This function is a private helper used by the `inOrderTraversal` method to perform an in-order traversal of the binary search tree. It takes `node` as a parameter, representing the current node being examined. The function follows the left -> root -> right order for traversal. It first recursively traverses the left subtree by calling itself with `node->left`. Then it prints the value of the current node, `node->data`, and finally, it recursively traverses the right subtree by calling itself with `node->right`.

- `preOrderTraversalRecursive(Node* node) const`: This function is a private helper used by the `preOrderTraversal` method to perform a pre-order traversal of the binary search tree. It takes `node` as a parameter, representing the current node being examined. The function follows the root -> left -> right order for traversal. It first prints the value of the current node, `node->data`, and then recursively traverses the left subtree by calling itself with `node->left`. Finally, it recursively traverses the right subtree by calling itself with `node->right`.

- `postOrderTraversalRecursive(Node* node) const`: This function is a private helper used by the `postOrderTraversal` method to perform a post-order traversal of the binary search tree. It takes `node` as a parameter, representing the current node being examined. The function follows the left -> right -> root order for traversal. It first

Regenerate