#include using namespace std; // Node class class Node { public: int data; Node* left; Node* right; Node(int value) { data = value; left = nullptr; right = nullptr; } }; // Binary Search Tree class class BST { private: Node* root; // Recursive helper function for inserting a value Node* insertRecursive(Node* currentNode, int value) { if (currentNode == nullptr) { return new Node(value); } if (value < currentNode->data) { currentNode->left = insertRecursive(currentNode->left, value); } else if (value > currentNode->data) { currentNode->right = insertRecursive(currentNode->right, value); } return currentNode; } // Recursive helper function for searching a value bool searchRecursive(Node* currentNode, int value) { if (currentNode == nullptr) { return false; } if (value == currentNode->data) { return true; } else if (value < currentNode->data) { return searchRecursive(currentNode->left, value); } else { return searchRecursive(currentNode->right, value); } } // Recursive helper function for in-order traversal void inorderTraversalRecursive(Node* currentNode) { if (currentNode != nullptr) { inorderTraversalRecursive(currentNode->left); cout << currentNode->data << " "; inorderTraversalRecursive(currentNode->right); } } public: BST() { root = nullptr; } // Public method to insert a value into the BST void insert(int value) { root = insertRecursive(root, value); } // Public method to search for a value in the BST bool search(int value) { return searchRecursive(root, value); } // Public method to perform in-order traversal of the BST void inorderTraversal() { inorderTraversalRecursive(root); cout << endl; } }; // Example usage int main() { BST tree; tree.insert(10); tree.insert(6); tree.insert(15); tree.insert(3); tree.insert(8); tree.insert(20); tree.inorderTraversal(); // Prints: 3 6 8 10 15 20 cout << "Is 8 present in the tree? " << (tree.search(8) ? "Yes" : "No") << endl; // Prints: Yes cout << "Is 17 present in the tree? " << (tree.search(17) ? "Yes" : "No") << endl; // Prints: No return 0; }