```
#include <iostream>
#include <algorithm> // For swap function
using namespace std;

class Node {
public:
    int value;
    Node* left;
    Node* right;

    Node(int key) {
        value = key;
        left = right = nullptr;
    }
};

class BinarySearchTree {
public:
    Node* root;

    BinarySearchTree() {
        root = nullptr;
    }

    Node* insert(Node* root, int key) {
        if (root == nullptr)
            return new Node(key);
        if (key < root->value)
            root->left = insert(root->left, key);
        else
            root->right = insert(root->right, key);
    return root;
}

int height(Node* root) {
    if (root == nullptr)
        return 0;
    int leftHeight = height(root->left);
    int rightHeight = height(root->right);
    return max(leftHeight, rightHeight) + 1;
}

int min_value(Node* root) {
    if (root == nullptr) {
        cout << "Tree is empty.\n";
        return -1; // or some sentinel value
    }
    while (root->left != nullptr)
        root = root->left;
    return root->value;
}

void swap_children(Node* root) {
    if (root == nullptr)
        return;
    swap(root->left, root->right);
    swap_children(root->left);
    swap_children(root->right);
}

Node* search(Node* root, int key) {
    if (root == nullptr || root->value == key)
```

```
            return root;
        if (key < root->value)
            return search(root->left, key);
        return search(root->right, key);
    }

    void inorder(Node* root) {
        if (root == nullptr)
            return;
        inorder(root->left);
        cout << root->value << " ";
        inorder(root->right);
    }
};

int main() {
    BinarySearchTree bst;
    int choice, value;
    Node* result = nullptr;



    do {
        cout << "\n### Binary Search Tree (BST) Menu ###";
        cout << "\nMenu:\n";
        cout << "1. Insert a Node\n";
        cout << "2. Find Height of Tree\n";
        cout << "3. Find Minimum Value\n";
        cout << "4. Swap Children\n";
        cout << "5. Search for a Value\n";
        cout << "6. Display In-order Traversal\n";
cout << "7. Exit\n";
cout << "Enter your choice: ";
cin >> choice;

switch (choice) {
case 1:
    cout << "Enter a value to insert: ";
    cin >> value;
    bst.root = bst.insert(bst.root, value);
    break;

case 2:
    cout << "Height of the tree: " << bst.height(bst.root) - 1 << endl;
    break;

case 3:
    {
        int minVal = bst.min_value(bst.root);
        if (minVal != -1)
            cout << "Minimum value in the tree: " << minVal << endl;
    }
    break;

case 4:
    bst.swap_children(bst.root);
    cout << "Children swapped at every node.\n";
    break;

case 5:
    cout << "Enter a value to search: ";
    cin >> value;
```

```
            result = bst.search(bst.root, value);
            if (result) {
                cout << "Found: " << result->value << endl;
            } else {
                cout << "Value not found in the tree.\n";
            }
            break;

        case 6:
            cout << "In-order traversal: ";
            bst.inorder(bst.root);
            cout << endl;
            break;

        case 7:
            cout << "Exiting the program.\n";
            break;

        default:
            cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 7);

    return 0;
}
```