Lab Guide 7: The Adventure of Little Fox with Binay Search Tree

Introduction:

In this lab, we will embark on an adventure with Little Fox, a curious explorer trapped in the depths of a forest. Little Fox must navigate through a tree's branches to find its way back to freedom. Along the way, we will learn about binary search trees and how to perform various operations such as adding, removing, and traversing nodes.

Objective:

The objective of this lab is to reinforce your understanding of binary search trees and their operations by applying them to a real-world scenario.

Background Story:

Once upon a time, in the heart of a vibrant forest filled with colorful birds, there lived a cheerful fox named Little Fox. Little Fox was known as a brave explorer who always sought to venture beyond the forest's borders. However, one day, on its return journey, Little Fox got lost and found itself trapped on a high tree branch. To escape this predicament, Little Fox must find the correct path through the branches.

Tasks:

1. **Implementing a Binary Search Tree:**
   - Begin by implementing a binary search tree data structure in Java.
   - Define a TreeNode class with fields for data, left child, and right child.(It is defined)
   - Implement methods to add nodes, remove nodes, and perform inorder traversal.

2. **Little Fox's Adventure:**
   - Simulate Little Fox's adventure using the binary search tree.
   - Generate a binary search tree representing the branches of the tree where Little Fox is trapped.
   - Allow Little Fox to choose a random branch (node) to climb, symbolizing its attempt to find the path to freedom.

3. **Binary Search Tree Operations:**
   - **addNodeRecursive(root, data)**: Special recursive method for adding a new node to the tree.
     - Hint: If the data we want to add is less than the root's data, got to the left subtree. If the data we want to add is greater than the root's data, got to the right subtree
   - **removeNodeRecursive(root, data)**: Special recursive method for removing a specific node from the tree.
   - Implement the **inorderTraversal** method to traverse the binary search tree in inorder fashion.
   - Implement **minValue(TreeNode root)** to find the minimum value in a binary search tree and using this method you will get the minimum value at the leftmost bottom node of a tree. Using this method, you can perform operations to find the minimum value on the tree.

4. **Example Outputs**

```
Binary search tree (inorder traversal) showing the path of the
fox cub
1 3 5 7 8

No hole found. Little Fox couldn't find a way out.

Binary search tree (inorder traversal) showing the path of the
little fox:
1 3 5 7 8
```

```
Binary search tree (inorder traversal) showing the path of the
fox cub
1 3 5 7 8

Little Fox found a hole and climbed into it: 7. It's free!

Binary search tree (inorder traversal) showing the path of the
little fox:
1 3 5 8
```