Lab Assignment: Binary Search Tree Operations

Objective

To implement a Binary Search Tree (BST) in C and perform fundamental operations on it.

Problem Statement

Design and implement a Binary Search Tree (BST) with the following operations:

- Insert a node
- Delete a node
- Find a node (search)
- Find minimum value
- Find maximum value
- Count total number of nodes
- Find the height of the tree
- Count the number of leaf nodes

Structure Definition and Function Signatures

```
struct Node {
int data;
struct Node* left;
struct Node* right;
};

// Creation and insertion
struct Node* createNode(int value);
struct Node* insert(struct Node* root, int value);

// Deletion
struct Node* deleteNode(struct Node* root, int value);

// Search
struct Node* search(struct Node* root, int key);

// Find minimum and maximum
struct Node* findMin(struct Node* root);
struct Node* findMax(struct Node* root);

// Utility functions
int countNodes(struct Node* root);
int findHeight(struct Node* root);
int countLeafNodes(struct Node* root);
```

Details

- The BST must maintain the property: [Left subtree values ¡ Root value ¡ Right subtree values]
- `insert` should place a new node in its correct position.

- `deleteNode` should correctly handle the three cases:
  1. Deleting a leaf node
  2. Deleting a node with one child
  3. Deleting a node with two children (use inorder successor or predecessor)
- `findMin` and `findMax` should traverse to the extreme left and right nodes, respectively.
- `countNodes` should return the total number of nodes.
- `findHeight` should return the height of the tree (maximum depth).
- `countLeafNodes` should return the total number of leaf nodes.

Expected Output

The program should allow creation and manipulation of a BST and produce correct results for all the above operations.