Tree Terminologies code

Please go through the following tree-related terms. Each one is implemented in the code you have. Understand what each term means and how it's derived from the binary tree structure.

1. **Root Node**

2. **Parent and Child**

3. **Siblings**

4. **Leaf Nodes**

5. **Internal Nodes**

6. **Degree of Node**

7. **Height of the Tree**

8. **Level of Node**

9. **Depth of a Node**

10. **Subtree**

Code Structure

```
#include <iostream>
using namespace std;

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

    Node(int value) {
```

```
        data = value;
        left = right = nullptr;
    }
};

// Print root node
void printRoot(Node* root) {
    if (root != nullptr)
        cout << "Root Node: " << root->data << endl;
}

// Print parent-child relationships
void printParentChild(Node* root) {
    if (root == nullptr) return;

    if (root->left != nullptr)
        cout << "Parent: " << root->data << " -> Left Child: " << root->left->data << endl;
    if (root->right != nullptr)
        cout << "Parent: " << root->data << " -> Right Child: " << root->right->data << endl;

    printParentChild(root->left);
    printParentChild(root->right);
}

// Print sibling pairs
void printSiblings(Node* root) {
    if (root == nullptr) return;

    if (root->left && root->right)
        cout << "Siblings: " << root->left->data << " & " << root->right->data << endl;

    printSiblings(root->left);
    printSiblings(root->right);
}

// Print leaf nodes
void printLeafNodes(Node* root) {
    if (root == nullptr) return;
    if (root->left == nullptr && root->right == nullptr) {
```

```
        cout << root->data << " ";
        return;
    }
    printLeafNodes(root->left);
    printLeafNodes(root->right);
}

// Print internal nodes
void printInternalNodes(Node* root) {
    if (root == nullptr) return;
    if (root->left != nullptr || root->right != nullptr)
        cout << root->data << " ";
    printInternalNodes(root->left);
    printInternalNodes(root->right);
}

// Print degree of each node
void printDegrees(Node* root) {
    if (root == nullptr) return;
    int degree = 0;
    if (root->left) degree++;
    if (root->right) degree++;
    cout << "Node " << root->data << " has degree: " << degree << endl;
    printDegrees(root->left);
    printDegrees(root->right);
}

// Find height of the tree
int findHeight(Node* root) {
    if (root == nullptr) return -1;
    int leftHeight = findHeight(root->left);
    int rightHeight = findHeight(root->right);
    return 1 + max(leftHeight, rightHeight);
}

// Print nodes at each level
void printLevels(Node* root, int level = 0) {
    if (root == nullptr) return;
    cout << "Node " << root->data << " is at level: " << level << endl;
```

```
    printLevels(root->left, level + 1);
    printLevels(root->right, level + 1);
}

// Print depth of a given node value
int findDepth(Node* root, int key, int depth = 0) {
    if (root == nullptr) return -1;
    if (root->data == key) return depth;

    int left = findDepth(root->left, key, depth + 1);
    if (left != -1) return left;

    return findDepth(root->right, key, depth + 1);
}

// Print all nodes in subtree rooted at given node
void printSubtree(Node* root) {
    if (root == nullptr) return;
    cout << root->data << " ";
    printSubtree(root->left);
    printSubtree(root->right);
}

// Find a node and print its subtree
bool findAndPrintSubtree(Node* root, int key) {
    if (root == nullptr) return false;
    if (root->data == key) {
        cout << "Subtree rooted at " << key << ": ";
        printSubtree(root);
        cout << endl;
        return true;
    }
    return findAndPrintSubtree(root->left, key) || findAndPrintSubtree(root->right, key);
}

// Main function
int main() {
    Node* root = new Node(1);
    root->left = new Node(2);
```

```
    root->right = new Node(3);
    root->left->left = new Node(4);
    root->left->right = new Node(5);

    cout << "\n== Tree Terminologies ==\n";

    printRoot(root);

    cout << "\nParent-Child Relationships:\n";
    printParentChild(root);

    cout << "\nSiblings:\n";
    printSiblings(root);

    cout << "\nLeaf Nodes: ";
    printLeafNodes(root);
    cout << endl;

    cout << "\nInternal Nodes: ";
    printInternalNodes(root);
    cout << endl;

    cout << "\nNode Degrees:\n";
    printDegrees(root);

    cout << "\nLevels of Nodes:\n";
    printLevels(root);

    cout << "\nHeight of Tree: " << findHeight(root) << endl;

    cout << "\nDepth of Node 5: " << findDepth(root, 5) << endl;

    cout << "\nSubtree rooted at node 2:\n";
    findAndPrintSubtree(root, 2);

    return 0;
}
```

Output

== Tree Terminologies ==

Root Node: 1

Parent-Child Relationships:
Parent: 1 -> Left Child: 2
Parent: 1 -> Right Child: 3
Parent: 2 -> Left Child: 4
Parent: 2 -> Right Child: 5

Siblings:
Siblings: 2 & 3
Siblings: 4 & 5

Leaf Nodes: 4 5 3

Internal Nodes: 1 2

Node Degrees:
Node 1 has degree: 2
Node 2 has degree: 2
Node 3 has degree: 0
Node 4 has degree: 0
Node 5 has degree: 0

Levels of Nodes:
Node 1 is at level: 0
Node 2 is at level: 1
Node 4 is at level: 2
Node 5 is at level: 2
Node 3 is at level: 1

Height of Tree: 2

Depth of Node 5: 2

Subtree rooted at node 2:
Subtree rooted at 2: 2 4 5