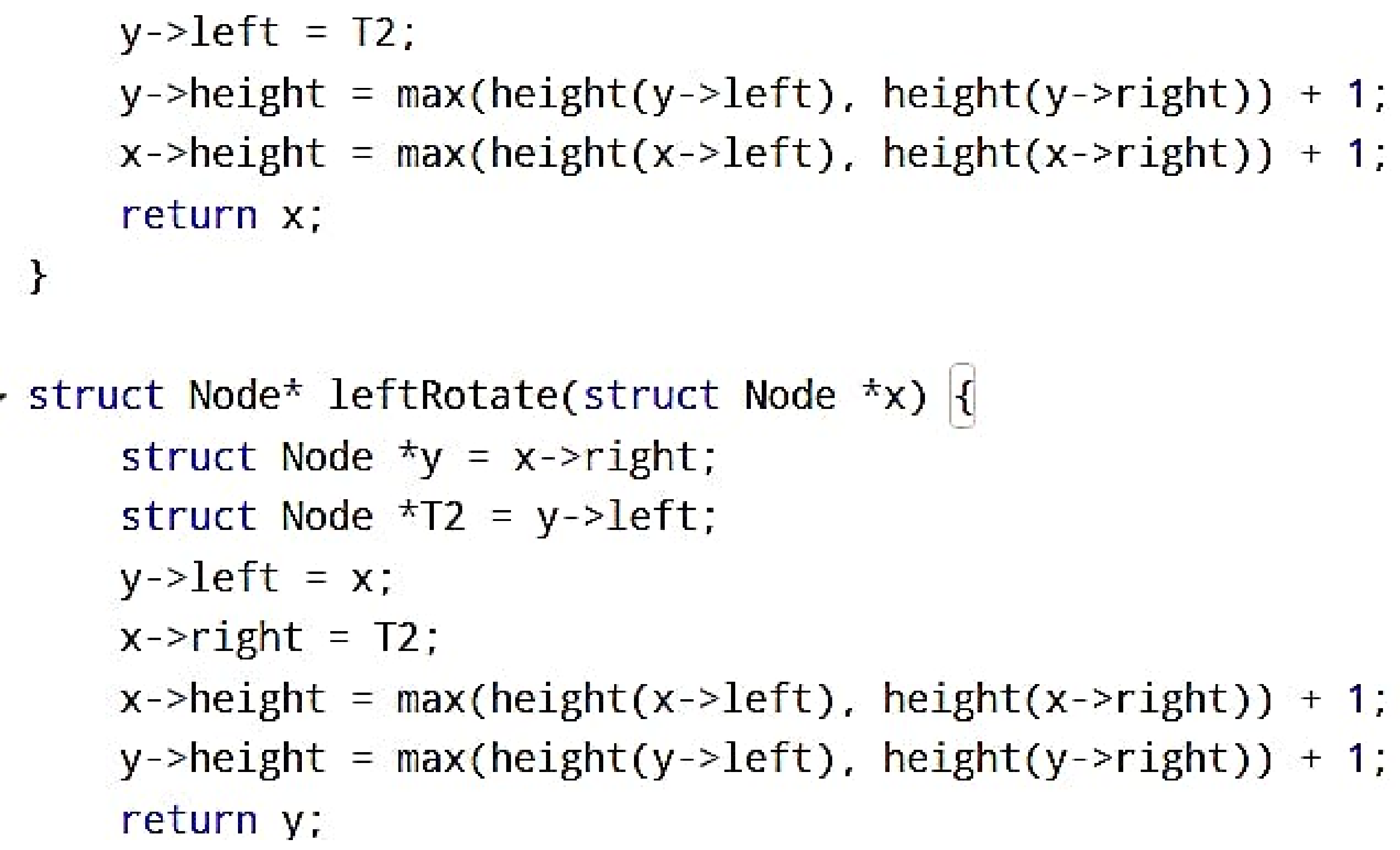
```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int key, height;
    struct Node *left, *right;
};

int height(struct Node *n) {
    return n ? n->height : 0;
}

int max(int a, int b) {
    return (a > b) ? a : b;
}

struct Node* newNode(int key) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->key = key;
    node->left = node->right = NULL;
    node->height = 1;
    return node;
}

struct Node* rightRotate(struct Node *y) {
    struct Node *x = y->left;
    struct Node *T2 = x->right;
    x->right = y;
```

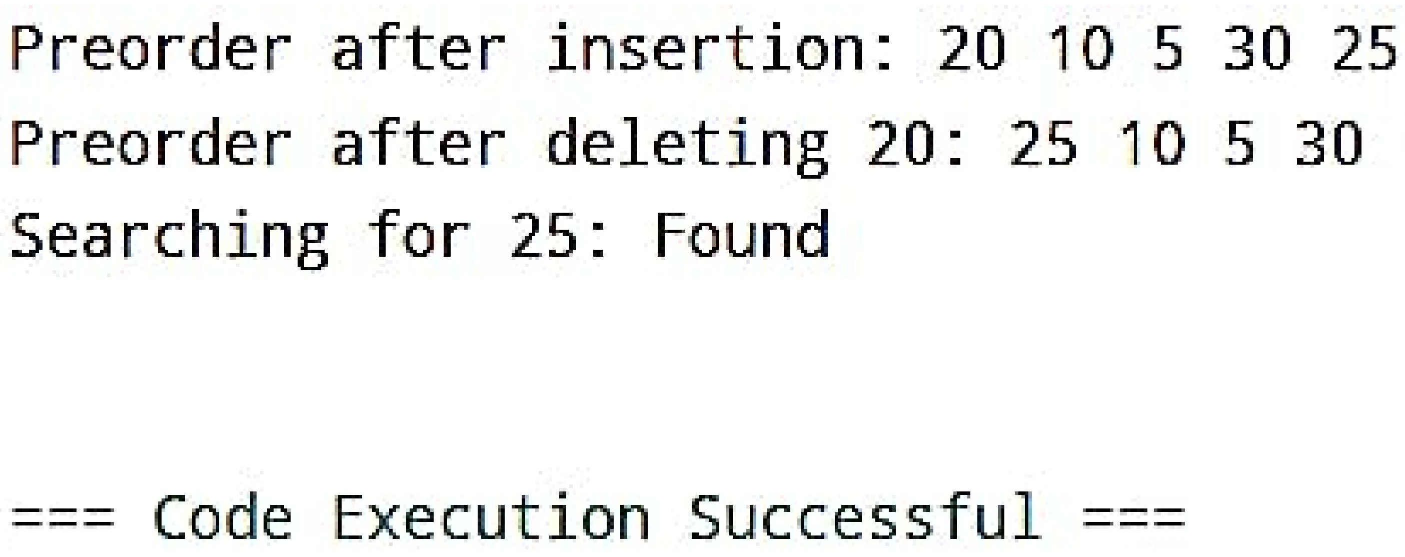


```c
    y->left = T2;
    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}

struct Node* leftRotate(struct Node *x) {
    struct Node *y = x->right;
    struct Node *T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;
    return y;
    if (balance > 1 && key < node->left->key)
        return rightRotate(node);
    if (balance < -1 && key > node->right->key)
        return leftRotate(node);
    if (balance > 1 && key > node->left->key) {
        node->left = leftRotate(node->left);
        return rightRotate(node);
    }
    if (balance < -1 && key < node->right->key) {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }
    return node;
}

struct Node* minValueNode(struct Node* node) {
    struct Node* current = node;
    while (current->left)
        current = current->left;
    return current;
}
```

```
    if (balance > 1 && getBalance(root->left) >= 0)
        return rightRotate(root);
    if (balance > 1 && getBalance(root->left) < 0) {
        root->left = leftRotate(root->left);
        return rightRotate(root);
    }
    if (balance < -1 && getBalance(root->right) <= 0)
        return leftRotate(root);
    if (balance < -1 && getBalance(root->right) > 0) {
        root->right = rightRotate(root->right);
        return leftRotate(root);
    }
    return root;
}

int search(struct Node* root, int key) {
    if (!root)
        return 0;
    if (key == root->key)
        return 1;
    else if (key < root->key)
        return search(root->left, key);
    else
        return search(root->right, key);
}
```

```
            ->right;
        if (!temp) {
            temp = root;
            root = NULL;
        } else
            *root = *temp;
        free(temp);
    } else {
        struct Node* temp = minValueNode(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key);
    }
}

if (!root) return root;

root->height = 1 + max(height(root->left), height(root->right
    ));
```

```c
void preOrder(struct Node *root) {
    if (root) {
        printf("%d ", root->key);
        preOrder(root->left);
        preOrder(root->right);
    }
}

int main() {
    struct Node *root = NULL;
    root = insert(root, 10);
    root = insert(root, 20);
    root = insert(root, 30);
    root = insert(root, 25);
    root = insert(root, 5);

    printf("Preorder after insertion: ");
    preOrder(root);

    root = deleteNode(root, 20);
    printf("\nPreorder after deleting 20: ");
    preOrder(root);

    int key = 25;
    printf("\nSearching for %d: %s\n", key, search(root,
        "Found" : "Not Found");
```



```
Preorder after insertion: 20 10 5 30 25
Preorder after deleting 20: 25 10 5 30
Searching for 25: Found


=== Code Execution Successful ===
```