```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int key;
    struct Node *left, *right;
    int height;
} Node;
int height(Node *n) {
    return n ? n->height : 0;
}
Node* createNode(int key) {
    Node *n = (Node*)malloc(sizeof(Node));
    n->key = key;
    n->left = n->right = NULL;
    n->height = 1;
    return n;
}
int max(int a, int b) {
    return (a > b) ? a : b;
}
Node* rightRotate(Node *y) {
    Node *x = y->left;
    Node *T2 = x->right;
    x->right = y;
    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;
}
Node* leftRotate(Node *x) {
    Node *y = x->right;
    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;
}
int getBalance(Node *n) {
    return n ? height(n->left) - height(n->right) : 0;
}
Node* insert(Node *node, int key) {
    if (!node) return createNode(key);
    if (key < node->key)
```

```
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else
        return node;
    node->height = 1 + max(height(node->left), height(node->right));
    int balance = getBalance(node);
    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;
}
Node* minValueNode(Node* node) {
    Node* current = node;
    while (current->left)
        current = current->left;
    return current;
}
Node* deleteNode(Node* root, int key) {
    if (!root) return root;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
        root->right = deleteNode(root->right, key);
    else {
        if (!root->left || !root->right) {
            Node *temp = root->left ? root->left : root->right;
            if (!temp) {
                temp = root;
                root = NULL;
            } else
                *root = *temp;
            free(temp);
        } else {
            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));
    int balance = getBalance(root);
    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;
}
Node* search(Node* root, int key) {
    if (!root || root->key == key) return root;
    if (key < root->key) return search(root->left, key);
    return search(root->right, key);
}
void inorder(Node *root) {
    if (root) {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}
int main() {
    Node *root = NULL;
    int keys[] = {20, 4, 15, 70, 50, 100, 80};
    for (int i = 0; i < 7; i++)
        root = insert(root, keys[i]);
    printf("After insertion: ");
    inorder(root); printf("\n");
    root = deleteNode(root, 70);
    printf("After deleting 70: ");
    inorder(root); printf("\n");
    int key = 15;
```

```
    printf("Searching %d: %sfound\n", key,
        search(root, key) ? "" : "not ");
    return 0;
}
```

C:\Users\upper\OneDrive\DATA STRUCTRES\Write a C program to imp

```
After insertion: 4 15 20 50 70 80 100
After deleting 70: 4 15 20 50 80 100
Searching 15: found

--------------------------------
Process exited after 0.1409 seconds with return value 0
Press any key to continue . . .
```