#include <stdio.h>

#include <stdlib.h>

struct Node {

int key;

struct Node \*left;

struct Node \*right;

int height;

};

int height(struct Node \*node) {

if (node == NULL)

return 0;

return node->height;

}

int getBalance(struct Node \*node) {

if (node == NULL)

return 0;

return height(node->left) - height(node->right);

}

struct Node \*newNode(int key) {

struct Node \*node = (struct Node \*)malloc(sizeof(struct Node));

node->key = key;

node->left = NULL;

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;

y->left = T2;

y->height = 1 + (height(y->left) > height(y->right) ? height(y->left) : height(y->right));

x->height = 1 + (height(x->left) > height(x->right) ? height(x->left) : height(x->right));

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 = 1 + (height(x->left) > height(x->right) ? height(x->left) : height(x->right));

y->height = 1 + (height(y->left) > height(y->right) ? height(y->left) : height(y->right));

return y;

}

struct Node \*insert(struct Node \*node, int key) {

if (node == NULL)

return newNode(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 + (height(node->left) > height(node->right) ? 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;

}

struct Node \*minValueNode(struct Node \*node) {

struct Node \*current = node;

while (current->left != NULL)

current = current->left;

return current;

}

struct Node \*deleteNode(struct Node \*root, int key) {

if (root == NULL)

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 == NULL || root->right == NULL) {

struct Node \*temp = root->left ? root->left : root->right;

if (temp == NULL) {

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 == NULL)

return root;

root->height = 1 + (height(root->left) > height(root->right) ? 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;

}

struct Node \*search(struct Node \*root, int key) {

if (root == NULL || root->key == key)

return root;

if (key < root->key)

return search(root->left, key);

else

return search(root->right, key);

}

void inOrderTraversal(struct Node \*root) {

if (root != NULL) {

inOrderTraversal(root->left);

printf("%d ", root->key);

inOrderTraversal(root->right);

}

}

int main() {

struct Node \*root = NULL;

root = insert(root, 10);

root = insert(root, 20);

root = insert(root, 30);

root = insert(root, 40);

root = insert(root, 50);

root = insert(root, 25);

printf("In-order traversal of the AVL tree: ");

inOrderTraversal(root);

printf("\n");

int keyToDelete = 30;

root = deleteNode(root, keyToDelete);

printf("In-order traversal after deleting %d: ", keyToDelete);

inOrderTraversal(root);

printf("\n");

int keyToSearch = 40;

struct Node \*searchResult = search(root, keyToSearch);

if (searchResult)

printf("%d found in the AVL tree.\n", keyToSearch);

else

printf("%d not found in the AVL tree.\n", keyToSearch);

return 0;

}

**OUTPUT:**

**In-order traversal of the AVL tree: 10 20 25 30 40 50**

**In-order traversal after deleting 30: 10 20 25 40 50**

**40 found in the AVL tree.**

**--------------------------------**

**Process exited after 0.03257 seconds with return value 0**

**Press any key to continue . . .**