#include <stdio.h>

#include <stdlib.h>

struct Node {

int value;

struct Node\* left;

struct Node\* right;

int height;

};

int height(struct Node\* node) {

if (node == NULL) return 0;

return node->height;

}

int max(int a, int b) {

return (a > b) ? a : b;

}

struct Node\* newNode(int value) {

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

node->value = value;

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;

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;

}

int getBalance(struct Node\* node) {

if (node == NULL) return 0;

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

}

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

if (node == NULL) return newNode(value);

if (value < node->value) node->left = insert(node->left, value);

else if (value > node->value) node->right = insert(node->right, value);

else return node;

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

int balance = getBalance(node);

if (balance > 1 && value < node->left->value) return rightRotate(node);

if (balance < -1 && value > node->right->value) return leftRotate(node);

if (balance > 1 && value > node->left->value) {

node->left = leftRotate(node->left);

return rightRotate(node);

}

if (balance < -1 && value < node->right->value) {

node->right = rightRotate(node->right);

return leftRotate(node);

}

return node;

}

void inorder(struct Node\* root) {

if (root != NULL) {

inorder(root->left);

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

inorder(root->right);

}

}

int main() {

struct Node\* root = NULL;

int value, choice;

while (1) {

printf("\n1. Insert value\n2. Inorder traversal\n3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("Inorder: ");

inorder(root);

printf("\n");

break;

case 3:

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

1. Insert value

2. Inorder traversal

3. Exit

Enter your choice: 1

Enter value to insert: 45

1. Insert value

2. Inorder traversal

3. Exit

Enter your choice: 1

Enter value to insert: 78

1. Insert value

2. Inorder traversal

3. Exit

Enter your choice: 1

Enter value to insert: 59

1. Insert value

2. Inorder traversal

3. Exit

Enter your choice: 1

Enter value to insert: 54

1. Insert value

2. Inorder traversal

3. Exit

Enter your choice: 2

Inorder: 45 54 59 78

1. Insert value

2. Inorder traversal

3. Exit

Enter your choice: 3