Day-10

1. #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, \*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, \*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 \*N) { return N ? height(N->left) - height(N->right) : 0; }

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

if (!node) 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 + 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;

}

void preOrder(struct Node \*root) {

if (root) {

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

preOrder(root->left);

preOrder(root->right);

}

}

int main() {

struct Node \*root = NULL;

int keys[] = {10, 20, 30, 40, 50, 25};

for (int i = 0; i < 6; i++) root = insert(root, keys[i]);

printf("Preorder traversal of the constructed AVL tree is: ");

preOrder(root);

return 0;

}

Output:

