Q)Binary Search Tree Traversal

#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

};

struct TreeNode \*createNode(int key) {

struct TreeNode \*newNode = (struct TreeNode \*)malloc(sizeof(struct TreeNode));

newNode->val = key;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

struct TreeNode \*insert(struct TreeNode \*root, int key) {

if (root == NULL)

return createNode(key);

if (key < root->val)

root->left = insert(root->left, key);

else if (key > root->val)

root->right = insert(root->right, key);

return root;

}

void inorderTraversal(struct TreeNode \*root) {

if (root != NULL) {

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

}

void preorderTraversal(struct TreeNode \*root) {

if (root != NULL) {

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

}

void postorderTraversal(struct TreeNode \*root) {

if (root != NULL) {

postorderTraversal(root->left);

postorderTraversal(root->right);

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

}

}

void display(struct TreeNode \*root) {

printf("In-order traversal: ");

inorderTraversal(root);

printf("\nPre-order traversal: ");

preorderTraversal(root);

printf("\nPost-order traversal: ");

postorderTraversal(root);

printf("\n");

}

int main() {

struct TreeNode \*root = NULL;

int keys[] = {50, 30, 20, 40, 70, 60, 80};

int i, n;

n = sizeof(keys) / sizeof(keys[0]);

for (i = 0; i < n; i++) {

root = insert(root, keys[i]);

}

display(root);

return 0;

}

