Tree Traversals

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* createNode(int data) {

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

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

void inorderTraversal(struct Node\* root) {

if (root == NULL)

return;

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

void preorderTraversal(struct Node\* root) {

if (root == NULL)

return;

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

void postorderTraversal(struct Node\* root) {

if (root == NULL)

return;

postorderTraversal(root->left);

postorderTraversal(root->right);

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

}

int main() {

struct Node\* root = createNode(1);

root->left = createNode(2);

root->right = createNode(3);

root->left->left = createNode(4);

root->left->right = createNode(5);

printf("Inorder traversal: ");

inorderTraversal(root);

printf("\n");

printf("Preorder traversal: ");

preorderTraversal(root);

printf("\n");

printf("Postorder traversal: ");

postorderTraversal(root);

printf("\n");

return 0;

}

OUTPUT

Inorder traversal: 4 2 5 1 3

Preorder traversal: 1 2 4 5 3

Postorder traversal: 4 5 2 3 1

--------------------------------

Process exited after 0.01631 seconds with return value 0

Press any key to continue . . .