AIM Binary Tree Traversals

CODE

// Tree traversal in C

#include <stdio.h>

#include <stdlib.h>

struct node {

int item;

struct node\* left;

struct node\* right;

};

// Inorder traversal

void inorderTraversal(struct node\* root) {

if (root == NULL) return;

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

// preorderTraversal traversal

void preorderTraversal(struct node\* root) {

if (root == NULL) return;

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

// postorderTraversal traversal

void postorderTraversal(struct node\* root) {

if (root == NULL) return;

postorderTraversal(root->left);

postorderTraversal(root->right);

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

}

// Create a new Node

struct node\* createNode(value) {

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

newNode->item = value;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Insert on the left of the node

struct node\* insertLeft(struct node\* root, int value) {

root->left = createNode(value);

return root->left;

}

// Insert on the right of the node

struct node\* insertRight(struct node\* root, int value) {

root->right = createNode(value);

return root->right;

}

int main() {

struct node\* root = createNode(1);

insertLeft(root, 12);

insertRight(root, 9);

insertLeft(root->left, 5);

insertRight(root->left, 6);

printf("Inorder traversal \n");

inorderTraversal(root);

printf("\nPreorder traversal \n");

preorderTraversal(root);

printf("\nPostorder traversal \n");

postorderTraversal(root);

}

**ALGORITHM**

**Inorder traversal**

First, visit all the nodes in the left subtree

Then the root node

Visit all the nodes in the right subtree

**Preorder traversal**

Visit root node

Visit all the nodes in the left subtree

Visit all the nodes in the right subtree

**Postorder traversal**

Visit all the nodes in the left subtree

Visit all the nodes in the right subtree

Visit the root node

# Output

