14. program to implement the Tree Traversals (Inorder, Preorder, Postorder):

#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);

}

//PREORDER TRAVERSAL

void preorderTraversal(struct node\*root)

{

if (root==NULL)

return ;

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

//POSTORDER 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 side 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->right,6);

printf("Inorder traversal\n");

inorderTraversal(root);

printf("Preorder traversal\n");

preorderTraversal(root);

printf("Postorder traversal\n");

postorderTraversal(root);

return 0;

}

