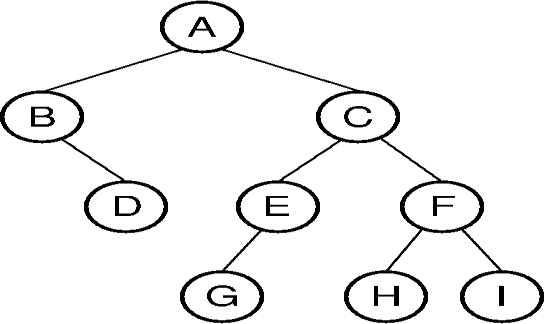
**DAA-LAB , V-SEM CSE**

**EXPR NO. -7**

**TREE TRAVESRSALS**

## OBJECTIVE:

Perform various tree traversal algorithms for a given tree.



## RESOURCES:

Dev C++

## PROGRAM LOGIC:

Traversal is a process to visit all the nodes of a tree and may print their values too.

Ex :-

Inorder(tree)

* + 1. Traverse the left subtree, i.e., call Inorder(left-subtree)
    2. Visit the root.
    3. Traverse the right subtree, i.e., call Inorder(right-subtree)

Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left-subtree)
2. Traverse the right subtree, i.e., call Postorder(right-subtree)
3. Visit the root.

Preorder(tree)

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left-subtree)
3. Traverse the right subtree, i.e., call Preorder(right-subtree)

## PROCEDURE:

* + 1. Create: Open Dev C++, write a program after that save the program with .c extension.
    2. Compile: Alt + F9
    3. Execute: Ctrl + F10

## SOURCE CODE:

#include<stdio.h> #include<stdlib.h> typedef struct treeNode{

int data; structtreeNode \*left; structtreeNode \*right;

}treeNode;

treeNode\* FindMin(treeNode \*node){

if(node==NULL){/\* There is no element in the tree \*/ return NULL;

}

if(node->left) /\* Go to the left sub tree to find the min element \*/ returnFindMin(node->left);

else

return node;}

treeNode \* insert(treeNode \*node,int data){ if(node==NULL){

treeNode \*temp;

temp = (treeNode \*)malloc(sizeof(treeNode)); temp -> data = data;

temp -> left = temp -> right = NULL; return temp;

}

if(data >(node->data)) {

node->right = insert(node->right,data);

}

else if(data < (node->data)){

node->left = insert(node->left,data);

}

/\* Else there is nothing to do as the data is already in the tree. \*/ return node;

}

treeNode \* deletion(treeNode \*node, int data){ treeNode \*temp;

if(node==NULL){

printf("Element Not Found");

}

else if(data < node->data){

node->left = deletion(node->left, data);

}

else if(data > node->data){

node->right = deletion(node->right, data);

}

else{

/\* Now We can delete this node and replace with either minimum elementin the right sub tree or maximum element in the left subtree \*/

if(node->right && node->left){

/\* Here we will replace with minimum element in the right sub tree \*/ temp = FindMin(node->right);

node -> data = temp->data;

/\* As we replaced it with some other node, we have to delete that node \*/ node -> right = deletion(node->right,temp->data);

}

else{

/\* If there is only one or zero children then we can directlyremove it

from the tree and connect its parent to its child \*/

temp = node;

if(node->left == NULL)

node = node->right; else if(node->right == NULL)

node = node->left;

free(temp); /\* temp is longer required \*/

}

}

return node;

}

treeNode \* search(treeNode \*node, int data){ if(node==NULL){/\* Element is not found \*/

return NULL;

}

if(data > node->data){ /\* Search in the right sub tree. \*/ return search(node->right,data);

}

else if(data < node->data){ /\* Search in the left sub tree. \*/ return search(node->left,data);

}

else{ /\* Element Found \*/ return node;

}

}

voidinorder(treeNode \*node){ if(node!=NULL) {

inorder(node->left); printf("%d ",node->data); inorder(node->right);

}

else return;

}

void preorder(treeNode \*node){ if(node!=NULL){

printf("%d ",node->data); preorder(node->left);

preorder(node->right);

}

else return;

}

Voidpostorder(treeNode \*node){ if(node!=NULL){

postorder(node->left); postorder(node->right); printf("%d ",node->data);

}

else return;

}

void main(){

treeNode \*t,\*root = NULL; intch, elt;

do {

printf("\n ### Binary Search Tree Operations ###"); printf("\n Press 1-Creation of BST");

printf("\n 2-deleting "); printf("\n 3-searching ");

printf("\n 4-Traverse in Inorder"); printf("\n 5-Traverse in Preorder"); printf("\n 6-Traverse in Postorder"); printf("\n 7-Exit\n");

printf("\n enter your choice ");

scanf("%d", &ch);

switch (ch) { case 1:

printf("enter element to be inserted"); scanf("%d", &elt);

root = insert(root, elt); break;

case 2:

case 3:

printf("enter element to be deleted"); scanf("%d",&elt);

deletion(root,elt); break;

printf("enter element to be search"); scanf("%d",&elt); t=search(root,elt);

if(t==NULL)

printf("element NOT found");

case 4:

break;

printf("\n BST Traversal in INORDER\n"); inorder(root);

break;

case 5:

printf("\n BST Traversal in PREORDER \n"); preorder(root);

break;

case 6:

printf("\n BST Traversal in POSTORDER \n"); postorder(root);

break;

case 7: printf("\n\n Terminating \n\n"); break;

default:

printf("\n\nInvalid Option !!! Try Again !! \n\n"); break;

}

} while (ch!= 7);

}

## INPUT/ OUTPUT



* 1. **LAB VIVA QUESTIONS:**
     1. Define binary tree.
     2. List different tree traversals.
     3. Explain inorder travels with example.
     4. Explain preorder travels with example.
     5. Explain postorder travels with example.