AIM:-

Aassignment 11

Create a binary search tree( BST)

a)Perform recursive inorder , preorder and postorder traversals.

OBJECTIVE:-

Traversing a tree means visiting every node in the tree. You might for instance want to add all the values in the tree or find the largest one. For all these operations, you will need to visit each node of the tree.

Theory :-

1. Uses of Inorder In case of binary search trees (BST), Inorder traversal gives nodes in non- decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal s reversed can be used.
2. Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expression on of an expression tree.
3. Postorder traversal is used to delete the tree.To delete a tree we must traverse all the nodes of the tree and delete them one by one,because before deleting the parent node we should delete its children nodes first.

Postorder traversal is also useful to get the postfix expression of an expression tree.

Algorithm :-

Sourcecode :-

#include<iostream>

using namespace std;

class BST {

struct node {

int data;

node\* left;

node\* right;

};

node\* root;

node\* makeEmpty(node\* t) {

if(t == NULL)

return NULL;

{

makeEmpty(t->left);

makeEmpty(t->right);

delete t;

}

return NULL;

}

node\* insert(int x, node\* t)

{

if(t == NULL)

{

t = new node;

t->data = x;

t->left = t->right = NULL;

}

else if(x < t->data)

t->left = insert(x, t->left);

else if(x > t->data)

t->right = insert(x, t->right);

return t;

}

void inorder(node\* t) {

if(t == NULL)

return;

inorder(t->left);

cout << t->data << " ";

inorder(t->right);

}

void preorder(node\* t) {

if(t == NULL)

return;

cout << t->data << " ";

preorder(t->left);

preorder(t->right);

}

void postorder(node\* t) {

if(t == NULL)

return;

postorder(t->left);

postorder(t->right);

cout << t->data << " ";

}

public:

BST() {

root = NULL;

}

~BST() {

root = makeEmpty(root);

}

void insert(int x) {

root = insert(x, root);

}

void display() {

cout<<"inorder"<<endl;

inorder(root);

cout<<endl;

cout<<"preorder"<<endl;

preorder(root);

cout<<endl<<"postorder"<<endl;

postorder(root);

cout << endl;

}

};

int main() {

BST t;

int n,ele;

cout<<"how many elements you want to insert in BST"<<endl;

cin>>n;

cout<<"start entering elements"<<endl;

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

{

cin>>ele;

t.insert(ele);

}

t.display();

return 0;

}

Output:-

how many elements you want to insert in BST

5

start entering elements

2

8

3

5

1

inorder

1 2 3 5 8

preorder

2 1 8 3 5

postorder

1 5 3 8 2

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