Experiment no : 4

#include <iostream>

using namespace std;

struct node {

int data;

node \*left;

node \*right;

};

class BST {

public:

node \*root;

BST() {

root=NULL;

}

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

node \*p =new node;

p->data=x;

p->left=NULL;

p->right=NULL;

if(t==NULL) {

t=p;

return t;

}

if(x<t->data) {

t->left=insert(t->left,p->data);

}

else {

t->right=insert(t->right,p->data);

}

}

void create() {

int n;

cout<<"Enter the number of elements: "<<endl;

cin>>n;

int x;

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

cout<<"Enter the values: "<<endl;

cin>>x;

root=insert(root,x);

}

}

void preorder(node \*t) {

if(t==NULL) {

return;

}

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

preorder(t->left);

preorder(t->right);

}

void inorder(node \*t) {

if(t==NULL) {

return;

}

inorder(t->left);

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

inorder(t->right);

}

void postorder(node \*t) {

if(t==NULL) {

return;

}

postorder(t->left);

postorder(t->right);

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

}

void display() {

cout<<"Preorder : ";

preorder(root);

cout<<"\nInorder : ";

inorder(root);

cout<<"\nPostorder : ";

postorder(root);

}

void min() {

node \*m=root;

while(m->left!=NULL) {

m=m->left;

}

cout<<"\nMinimun value : ";

cout<<m->data<<"\n";

}

void search()

{

node \*s=root;

int val;

cout<<"\nEnter the element to serach: ";

cin>>val;

int flag=0;

while(s!=NULL)

{

if(val==s->data)

{

cout<<s->data;

flag=1;

}

if(val<s->data)

{

s=s->left;

}

else

{

s=s->right;

}

}

if(flag==1)

{

cout<<"\nElement found\n";

}

else

{

cout<<"Element not found\n";

}

}

node \*mirror(node \*root)

{

node \*p;

if(root!=NULL)

{

p=root->left;

root->left=mirror(root->right);

root->right=mirror(p);

}

return(root);

}

int height(node \*root)

{

if(root==NULL)

{

return 0;

}

else if(root->left==NULL && root->right==NULL)

{

return 0;

}

else

{

return max(height(root->left),height(root->right))+1;

}

}

int max(int x,int y)

{

if(x>=y)

{

return x;

}

else

{

return y;

}

}

};

int main()

{

BST b;

int ch;

do{

cout<<"\n1.To create a tree \n";

cout<<"2.To display a tree \n";

cout<<"3.To find minimun element in the tree \n";

cout<<"4.To search an element in the tree \n";

cout<<"5.To create mirror of the tree \n";

cout<<"6.To display mirror of the tree \n";

cout<<"7.To display the height of the tree \n";

cout<<"8.To display the number of node in longest path of the tree \n";

cout<<"9.To exit the program\n";

cout<<"Enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

{

b.create();

break;

}

case 2:

{

b.display();

break;

}

case 3:

{

b.min();

break;

}

case 4:

{

b.search();

break;

}

case 5:

{

b.mirror(b.root);

cout<<"Mirror of tree is created\n";

break;

}

case 6:

{

b.display();

break;

}

case 7:

{

cout<<"\nHeight: "<<b.height(b.root);

break;

}

case 8:{

cout<<"\nCount: "<<b.height(b.root)+1;

break;

}

case 9:

{

cout<<"Exited successfully";

break;

}

default:

{

cout<<"Invalid choice!!";

break;

}

}

}while(ch!=9);

return 0;

}

Output :





