#include<iostream>

#include<stdlib.h>

#include<conio.h>

using namespace std;

struct BSTnode

{

int data;

BSTnode \*left,\*right;

};

class tree

{

public:

BSTnode \*root,\*temp;

public:

tree()

{

temp=NULL;

root=NULL;

}

void insert(int key);

void print\_inorder();

void inorder(BSTnode \*);

void print\_postorder();

void postorder(BSTnode \*);

void print\_preorder();

void preorder(BSTnode \*);

void del(int d);

void search(BSTnode \*root,int val);

};

void tree::insert(int key)

{

BSTnode \*newnode = new BSTnode;

BSTnode \*temp,\*parent=NULL;

newnode->data=key;

newnode->left=NULL;

newnode->right=NULL;

if(root==NULL)

{

root=newnode;

}

else

{

temp=root;

while(temp)

{

parent=temp;

if(newnode->data > temp->data)

{

temp=temp->right;

}

else

{

temp=temp->left;

}

}

if(newnode->data < parent ->data)

{

parent->left=newnode;

}

else

{

parent->right=newnode;

}

}

}

void tree::print\_inorder()

{

inorder(root);

}

void tree::inorder(BSTnode \*p)

{

if(p!=NULL)

{

if(p->left)

{

inorder(p->left);

}

cout<<" "<<p->data<<" ";

if(p->right)

{

inorder(p->right);

}

}

else

{

return;

}

}

void tree::print\_postorder()

{

postorder(root);

}

void tree::postorder(BSTnode \*p)

{

if(p!=NULL)

{

if(p->left)

{

postorder(p->left);

}

if(p->right)

{

postorder(p->right);

}

cout<<" "<<p->data<<" ";

}

else

{

return;

}

}

void tree::print\_preorder()

{

preorder(root);

}

void tree::preorder(BSTnode \*p)

{

if(p!=NULL)

{

cout<<" "<<p->data<<" ";

if(p->left)

{

preorder(p->left);

}

if(p->right)

{

preorder(p->right);

}

}

else

{

return;

}

}

void tree::del(int d)

{

bool found = false;

if(root==NULL)

{

cout<<" This Tree is empty! "<<endl;

return;

}

BSTnode \*curr;

BSTnode \*parent;

curr = root;

while(curr != NULL)

{

if(curr->data == d)

{

found = true;

break;

}

else

{

parent = curr;

if(d>curr->data) curr = curr->right;

else curr = curr->left;

}

}

if(!found)

{

cout<<" Data not found! "<<endl;

return;

}

if((curr->left == NULL && curr->right != NULL)|| (curr->left != NULL

&& curr->right == NULL))

{

if(curr->left == NULL && curr->right != NULL)

{

if(parent->left == curr)

{

parent->left = curr->right;

delete curr;

}

else

{

parent->right = curr->right;

delete curr;

}

}

else

{

if(parent->left == curr)

{

parent->left = curr->left;

delete curr;

}

else

{

parent->right = curr->left;

delete curr;

}

}

return;

}

if( curr->left == NULL && curr->right == NULL)

{

if(parent->left == curr)

{

parent->left = NULL;

}

else

parent->right = NULL;

delete curr;

return;

}

if (curr->left != NULL && curr->right != NULL)

{

BSTnode \*chkr;

chkr = curr->right;

if((chkr->left == NULL) && (chkr->right == NULL))

{

curr = chkr;

delete chkr;

curr->right = NULL;

}

else

{

if((curr->right)->left != NULL)

{

BSTnode \*lcurr;

BSTnode \*lcurrp;

lcurrp = curr->right;

lcurr = (curr->right)->left;

while(lcurr->left != NULL)

{

lcurrp = lcurr;

lcurr = lcurr->left;

}

curr->data = lcurr->data;

delete lcurr;

lcurrp->left = NULL;

}

else

{

BSTnode \*tmp;

tmp = curr->right;

curr->data = tmp->data;

curr->right = tmp->right;

delete tmp;

}

}

return;

}

}

void tree::search(BSTnode \*root,int val)

{

int num;

if((root==NULL))

{

cout<<"tree is empty";

}

if(root->data==val)

{

cout<<root;

num=root->data;

}

else

{

if(num<(root->data))

{

return search(root->left,val);

}

else

{

return search(root->right,val);

}

}

}

int main()

{

tree t;

int ch,n;

BSTnode \*k = new BSTnode;

k=NULL;

cout<<"\n1.Create binary search tree\n2.Inorder display\n3.Postorder display\n4.Preorder display\n5.Search\n6.Delete\n";

while(1)

{

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:cout<<"Enter a number:";

cin>>n;

t.insert(n);

break;

case 2:cout<<"Inorder traversal:";

t.print\_inorder();

Break;

case 3:cout<<"Postorder traversal:";

t.print\_postorder();

break;

case 4:cout<<"Preorder traversal:";

t.print\_preorder();

break;

case 5:cout<<"Enter a number to search:";

cin>>n;

t.search(k,n);

case 6:cout<<"Enter a number:";

cin>>n;

t.del(n);

break;

default:cout<<"Invalid choice";

}

}

}