#include<iostream>

#include<cmath>

using namespace std;

//===============================

struct node

{

int info;

node \*left;

node \*right;

};

//===================================

class cbt

{

node \*temp,\*temp1,\*temp2;

bool flag;

int count,num;

double totalheight,totalmaxnode,totalminnode;

public:

node \*root;

int key;

//================================================

cbt()

{

root=temp=temp1=temp2=0;

key=count=0;

totalheight=0,totalmaxnode=0,totalminnode=0;

num=1;

flag=true;

}

//=======================================================

void insert(node \*temp)

{

double totalheight=0,totalmaxnode=0,totalminnode=0;

num=1;

if(root==NULL)

{

root=new node;

root->info=key;

root->left=root->right=0;

temp=root;

count++;

return;

}

totalheight=height(root);

totalminnode=pow(2.0,totalheight);

totalmaxnode=pow(2.0,(totalheight+1))-1;

if(count==totalmaxnode)

{

while(temp->left!=NULL)

{

temp=temp->left;

}

temp->left=new node;

temp->left->info=key;

temp->left->left=temp->left->right=0;

count++;

return;

}

update(root,totalheight);

if(flag==true)

{

temp1->left=new node;

temp1->left->info=key;

temp1->left->left=temp1->left->right=0;

count++;

return;

}

if(flag==false)

{

temp1->right=new node;

temp1->right->info=key;

temp1->right->left=temp1->right->right=0;

count++;

return;}

}

//==================================================================================================================

void update(node \*temp,double ht)

{

if(temp->left==NULL)

{

temp1=temp;

flag=true;

num=2;

return;

}

if(temp->right==NULL)

{

temp1=temp;

flag=false;

num=2;

return;

}

if(ht>1)

update(temp->left,ht-1);

if(ht>1 && num==1)

update(temp->right,ht-1);

}

//================================================

void updatedelete(node \*temp,double ht)

{

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

{

temp2=temp;

flag=true;

num=2;

return;

}

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

{

temp2=temp;

flag=false;

}

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

{

num=2;

return;

}

if(ht>1)

update(temp->left,ht-1);

if(ht>1 && num==1)

update(temp->right,ht-1);

}

//==================================================

int height(node \*temp)

{

int step=0;

while(temp->left!=NULL)

{

temp=temp->left;

step++;

}

return step;

}

//===========================================

void inorder(node \*temp)

{

if(temp==NULL)

{ cout<<"THe tree is empty \n";

return;

}

if(temp->left!=NULL)

inorder(temp->left);

cout<<"\t"<<temp->info<<" ";

if(temp->right!=0)

inorder(temp->right);

return;

}

//==============================================================

void inorderdelete(node \*temp,int key)

{

if(temp->left!=NULL)

{inorderdelete(temp->left,key);

if (temp->info==key)

temp1=temp;

return;}

if(temp->right!=0)

inorderdelete(temp->right,key);

return;

}

//===============================================================

void deletes(node \*temp,int key)

{

if(temp==NULL)

{

cout<<" The tree is empty ."<<endl;

return;

}

inorderdelete(temp,key);

totalheight=height(root);

totalminnode=pow(2.0,totalheight);

totalmaxnode=pow(2.0,(totalheight+1))-1;

if(count==totalminnode)

{

while(temp->left!=NULL)

{

temp=temp->left;

}

temp1->info=temp->info;

count--;

temp=NULL;

}

if(count==totalmaxnode)

{

while(temp->right!=NULL)

{

temp=temp->right;

}

temp1->info=temp->info;

count--;

temp=NULL;

}

updatedelete(root,totalheight);

if(flag==true)

{

temp1->info=temp2->left->info;

temp2=NULL;

count--;

return;

}

if(flag==false)

{temp1->info=temp->right->info;

temp2=NULL;

count--;

return;

}

}

//=============================================================

};

void main()

{

cbt obj;

int a;

cout<<"insertion"<<endl;

for(int i=0;i<10;i++)

{

cin>>obj.key;

obj.insert(obj.root);

}

obj.inorder(obj.root);

cout<<"Enter info for deletion : ";

cin>>a;

obj.deletes(obj.root,a);

obj.inorder(obj.root);

system("pause");

}