//Project 25- BST Level order traversal with insertion and deletion functions.

//By-Akshay Mathur 189301109 Cse-B

#include<stdio.h>

#include<stdlib.h>

struct node{

struct node \*left;

struct node \*right;

int info;

};

typedef struct node NODE;

NODE \*createnode(int n)

{

NODE \*nn;

nn=(NODE\*)malloc(sizeof(NODE));

nn->left=nn->right=NULL;

nn->info=n;

return nn;

}

NODE \*createtree(NODE \*root)

{

NODE \*p,\*c;

int z=1,n,x;

printf("enter info part of root:\t");

scanf("%d",&x);

root=createnode(x);

while(z!=0)

{

printf("enter info part of next node:\t");

scanf("%d",&n);

p=c=root;

while(c!=NULL)

{

p=c;

if(n>p->info)

c=p->right;

else

c=p->left;

}

if(n>p->info)

p->right=createnode(n);

else

p->left=createnode(n);

printf("do you want to continue?:\t");

scanf("%d",&z);

}

return root;

}

void Insert(NODE \*root)

{

NODE \*p,\*c;

int n;

printf("enter info of new node:\t");

scanf("%d",&n);

p=c=root;

while(c!=NULL)

{

p=c;

if(n>p->info)

c=p->right;

else

c=p->left;

}

if(n>p->info)

p->right=createnode(n);

else

p->left=createnode(n);

}

void printLevel(NODE\* root, int level)

{

if(root == NULL)

return;

if(level == 1)

printf("%d ", root->info);

else if (level > 1)

{

printLevel(root->left, level-1);

printLevel(root->right, level-1);

}

}

int height(NODE\* node)

{

if (node==NULL)

return 0;

else

{

int lheight = height(node->left);

int rheight = height(node->right);

if (lheight > rheight)

return(lheight+1);

else

return(rheight+1);

}

}

void printLevelOrder(NODE\* root)

{

int h = height(root);

int i;

for(i=1; i<=h; i++)

printLevel(root, i);

}

NODE \*searchparent(NODE \*root,int x)

{

NODE \*p,\*c;

p=c=root;

while(x!=c->info)

{

p=c;

if(x>p->info)

c=p->right;

else

c=p->left;

}

return p;

}

NODE \*search(NODE \*root,int val)

{

while(root!=NULL&&root->info!=val)

{

if(val>root->info)

root=root->right;

else

root=root->left;

}

return root;

}

NODE \*inorderSuccessor(NODE \*root, NODE \*node)

{

NODE \*successor = NULL;

NODE \*current = root;

while(current->info != node->info)

{

if(current->info > node->info)

{

successor = current;

current= current->left;

}

else

current = current->right;

}

if(current!=NULL && current->right!=NULL)

{

successor=current->right;

while(successor->left!=NULL)

successor=successor->left;

}

return successor;

}

void swap(NODE \*a,NODE \*b)

{

NODE \*c;

c=createnode(0);

c->info=b->info;

b->info=a->info;

a->info=c->info;

}

NODE \*del(NODE \*root,NODE \*adr)

{

NODE \*p,\*q,\*r;

q=NULL;

if(adr==NULL)

return root;

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

{

p=searchparent(root,adr->info);

if(p->left==adr)

p->left=NULL;

else

p->right=NULL;

free(adr);

return root;

}

else if(adr->right!=NULL&&adr->left!=NULL)

{

q=inorderSuccessor(root,adr);

swap(adr,q);

if(adr->right==q)

adr->right=NULL;

else

{

r=adr->right;

while(r->left!=q)

r=r->left;

r->left=NULL;

}

free(q);

return root;

}

else

{

if(adr->left!=NULL)

{

swap(adr,adr->left);

adr->left=NULL;

free(adr->left);

return root;

}

else

{

swap(adr,adr->right);

adr->right=NULL;

free(adr->right);

return root;

}

}

}

void main()

{

int x,z;

NODE \*root,\*adr;

root=adr=NULL;

root=createtree(root);

do{

printf("enter choice\n1)Insert\n2.)Delete\n3.)Level order traverse\n4.)exit\n" );

scanf("%d",&z);

switch(z)

{

case 1:{Insert(root);

break;}

case 2:{printf("enter value to be deleted:\t");

scanf("%d",&x);

adr=search(root,x);

if(adr==NULL)

printf("info not found\n");

root=del(root,adr);

break;}

case 3:{printLevelOrder(root);

printf("\n");

break;}

}

}while(z!=4);

printLevelOrder(root);

}