#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<limits.h>

//int\_max ,int\_min

struct node{

int data;

struct node \*left;

struct node \*right;

};

struct node\* createnode(int val){

struct node\* temp=(struct node \*)malloc(sizeof(struct node));

temp->data=val;

temp->left=NULL;

temp->right=NULL;

return temp;

}

struct node\* insertbinarytree(struct node \*root,int val){

if(root==NULL){

root=createnode(val);

return root;

}

if(val>root->data){

root->right=insertbinarytree(root->right,val);

}else if(val<root->data){

root->left=insertbinarytree(root->left ,val);

}

return root;

}

struct node\* inordersuccessor(struct node\* root){

while(root->left!=NULL){

root=root->left;

}

return root;

}

struct node\* delete(struct node\* root,int val){

if(root==NULL){

return NULL;

}

if(val>root->data){

root->right=delete(root->right,val);

}

else if(val<root->data){

root->left=delete(root->left,val);

}

else{

//leaf node condition

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

return NULL;

}

//when left one child is null

if(root->left==NULL){

return root->right;

}

// when right one child is null

else if(root->right==NULL){

return root->left;

}

struct node\* is=inordersuccessor(root->right);

root->data=is->data;

root->right=delete(root->right,is->data);

}

return root;

}

void inordertraversal(struct node \*root){

if(root==NULL){

return;

}

inordertraversal(root->left);

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

inordertraversal(root->right);

}

void printinrange(struct node \*root,int k1,int k2){

if(root==NULL){

return;

}

printinrange(root->left,k1,k2);

if(root->data>=k1&&root->data<=k2){

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

}

printinrange(root->right,k1,k2);

}

bool searchinbst(struct node\* root,int val){

if(root==NULL){

return false;

}

if(root->data==val){

printf("found");

return true;

}

if(root->data<val){

return searchinbst(root->right,val);

}

else{

return searchinbst(root->left,val);

}

}

int max(int num1,int num2){

return (num1>num2)? num1:num2;

}

int countnode(struct node\* root){//find count node

if(root==NULL){

return 0;

}

int lc=countnode(root->left);

int rc=countnode(root->right);

return lc+rc+1;

}

int height(struct node\* root){//find height

if(root==NULL){

return 0;

}

int h1=height(root->left);

int h2=height(root->right);

return max(h1,h2)+1;

}

int diameter(struct node\* root)

{

int d1=diameter(root->left);

int h1=height(root->left);

int d2=diameter(root->right);

int h2=height(root->right);

int sd=h1+h2+1;

return max(sd,(max(d1,d2)));

}

void printkthlevel(struct node \*root,int k,int level){

if(root==NULL){

return;

}

if(k==level){

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

return;

}

printkthlevel(root->left,k,level+1);

printkthlevel(root->right,k,level+1);

}

int main(){

struct node \*root=NULL;

root =insertbinarytree(root,1);

root =insertbinarytree(root,2);

root =insertbinarytree(root,3);

root =insertbinarytree(root,4);

printf("/n");

inordertraversal(root);

printf("\n");

root=delete(root,2);

inordertraversal(root);

printf("\n-----");

printinrange(root,1,3);

searchinbst(root,1);

printf("%d",countnode(root));

printf("%d",height(root));

printkthlevel(root,3,1);

return 0;

}