//static auto \_ = [] ()

//{ios\_base::sync\_with\_stdio(false);cin.tie(nullptr);cout.tie(nullptr);return 0;}();

#include<bits/stdc++.h>

using namespace std;

class node{

public:

int data;

node \*left,\*right;

node(int val){

data = val;

left = NULL;

right = NULL;

}

};

node\* insertBst(node\* root, int val){

if(root==NULL){

return new node(val);

}

if(val<root->data){

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

}

else{

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

}

return root;

}

node\* search(node\* root, int val){

if(root==NULL) return NULL;

if(root->data<val){

return search(root->right,val);

}

else if(root->data==val){

return root;

}

else{

return search(root->left,val);

}

}

node\* max(node\* root){

if(root==NULL) return NULL;

if(root->right==NULL){

return root;

}

return max(root->right);

}

node\* min(node\* root){

if(root==NULL) return NULL;

if(root->left==NULL) return root;

return min(root->left);

}

node\* max\_prev(node\* root, node\* &prev){

if(root==NULL) return NULL;

if(root->right==NULL){

return root;

}

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

prev = root;

return root->right;

}

return max\_prev(root->right,prev);

}

node\* min\_prev(node\* root, node\* &prev){

if(root==NULL) return NULL;

if(root->left==NULL){

return root;

}

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

prev = root;

return root->left;

}

return min\_prev(root->left,prev);

}

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

if(root==NULL) return NULL;

if(root->data==val){

//case 1 for leaves node

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

return NULL;

}

//case 2 for one child

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

// node\* temp = root;

// root = root->right;

// temp = NULL;

return root->right;

}

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

// node\* temp = root;

// root = root->left;

// temp = NULL;

return root->left;

}

else{

//case 3 for two child

node\* prev;

prev = max(root->left);

root->data = prev->data;

root->left = delete\_node(root->left,prev->data);

return root;

}

}

else if(root->data>val){

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

}

else{

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

}

return root;

}

void inorder(node\* root){

if(root==NULL) return;

inorder(root->left);

cout<<root->data<<" ";

inorder(root->right);

}

int main(){

//6

// 5 1 3 4 2 7

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

}

node \*root = NULL;

root = insertBst(root,arr[0]);

for(int i=1;i<n;i++){

insertBst(root,arr[i]);

}

inorder(root);

cout<<endl;

root = delete\_node(root,3);

inorder(root);

// node\* prev;

// node\* m = min\_prev(root,prev);

// cout<<m->data<<endl;

// cout<<prev->data<<endl;

// if(search(root,2)!=NULL){

// cout<<"yes"<<endl;

// }

// else{

// cout<<"NO"<<endl;

// }

return 0;

}