/// insertion in bst

#include<bits/stdc++.h>

using namespace std;

struct BST

{

int data;

BST \*left,\*right;

};

int a[500000],cnt=7;

BST \*newnode(int value)

{

BST \*tmp=new BST;

tmp->data=value;

tmp->left=tmp->right=nullptr;

return tmp;

}

BST \*insert(BST \*root,int item)

{

if(root==nullptr)

return newnode(item);

else if(root->data<item)

root->right= insert(root->right,item);

else root->left=insert(root->left,item);

return root;

}

/\*void inorder(BST \*root)

{

if(root==nullptr)return;

inorder(root->left);

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

inorder(root->right);

}\*/

void print2DUtil(BST \*root, int space)

{

if (root == NULL)

return;

space += cnt;

print2DUtil(root->right, space);

cout<<endl;

for (int i = cnt; i < space; i++)

cout<<" ";

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

print2DUtil(root->left, space);

}

void print2D(BST \*root)

{

print2DUtil(root, 0);

}

int main()

{

int n,x;

printf("Enter the number of elements to add: ");

cin>>n;

BST \*root=nullptr;

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

{

cin>>a[i];

}

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

{

x=a[i];

root=insert(root,x);

}

print2D(root);

//inorder(root);

}

///deletion in bst

#include<bits/stdc++.h>

using namespace std;

struct BST

{

int data;

BST \*left,\*right;

};

int a[50000],n,i,x,cnt=7;

BST \*findminimum(BST \*root)

{

while(root->left)

root=root->left;

return root;

}

BST \*Delete(BST \*root,int item)

{

if(root==0)return root;

else if(item<root->data)

root->left=Delete(root->left,item);

else if(item>root->data)

root->right=Delete(root->right,item);

else

{

/// case 1:no child

if(root->left==0 && root->right==0)

{

delete(root);

root=0;

}

/// case 2:one child

else if(root->left==0)

{ BST \*tmp=root;

root=root->right;

delete tmp;

}

else if(root->right==0)

{BST \*tmp=root;

root=root->left;

delete tmp;}

/// case 3:2 child

else

{

BST \*tmp=findminimum(root->right);

root->data=tmp->data;

root->right=Delete(root->right,tmp->data);

}

}

return root;

}

BST \*insert(BST \*root,int item)

{

if(root==nullptr)

{root =new BST();

root->data=item;

root->left=root->right=nullptr;

}

else if(root->data<item)

root->right=insert(root->right,item);

else root->left=insert(root->left,item);

return root;

}

/\*void inorder(BST \*root)

{

if(root==nullptr)return;

inorder(root->left);

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

inorder(root->right);

}\*/

void print2DUtil(BST \*root, int space)

{

if (root == NULL)

return;

space += cnt;

print2DUtil(root->right, space);

cout<<endl;

for (int i = cnt; i < space; i++)

cout<<" ";

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

print2DUtil(root->left, space);

}

void print2D(BST \*root)

{

print2DUtil(root, 0);

}

int main()

{

printf("Enter the number of nodes to add: ");

cin>>n;

for(i=0;i<n;i++)cin>>a[i];

BST \*root=nullptr;

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

{

x=a[i];

root=insert(root,x);

}

print2D(root);

printf("Enter the node to delete: ");

cin>>x;

root=Delete(root,x);

//inorder(root);

cout<<"The tree after deleting node:"<<endl;

print2D(root);

}

/// find predecessor

#include<bits/stdc++.h>

using namespace std;

struct BST

{

int data;

BST \*left,\*right;

};

int a[2000000],cnt=5;

BST \*newnode(int item)

{

BST \*tmp=new BST;

tmp->data=item;

tmp->left=tmp->right=nullptr;

return tmp;

}

BST \*insert(BST \*root,int item)

{

if(root==nullptr)return newnode(item);

if(item<root->data)

root->left=insert(root->left,item);

else root->right=insert(root->right,item);

return root;

}

BST\* Find(BST \*root, int item) {

if(root == NULL) return NULL;

else if(root->data == item) return root;

else if(root->data < item) return Find(root->right,item);

else return Find(root->left,item);

}

BST \*findmaximum(BST \*root)

{

while(root->right!=0)

root=root->right;

return root;

}

BST \*findPredecessor(BST \*root,int item)

{

BST \*curr=Find(root,item);

if(curr==nullptr)return nullptr;

if(curr->left!=0) /// has left subtree

{

return findmaximum(curr->left);

}

else

{ /// no left subtree

BST \*pre=0,\*ancestor=root;

while(curr!=ancestor)

{

if(ancestor->data>curr->data)

ancestor=ancestor->left;

else

{

pre=ancestor;

ancestor=ancestor->right;

}

}

return pre;

}

}

void print2DUtil(BST \*root, int space)

{

if (root == NULL)

return;

space += cnt;

print2DUtil(root->right, space);

cout<<endl;

for (int i = cnt; i < space; i++)

cout<<" ";

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

print2DUtil(root->left, space);

}

void print2D(BST \*root)

{

print2DUtil(root, 0);

}

int main()

{

int n,i,x;

printf("Enter the number of elements to add: ");

cin>>n;

BST \*root=0;

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

{

cin>>a[i];

x=a[i];

root=insert(root,x);

}

print2D(root);

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

{

x=a[i];

BST \*pre= findPredecessor(root,x);

if(pre!=nullptr)

cout<<"Predecessor of node "<<x<<" is: "<<pre->data<<endl;

else cout<<"Predecessor doesn't exist for node "<<x<<endl;

}

return 0;

}

/// find successor

#include<bits/stdc++.h>

using namespace std;

struct BST

{

int data;

BST \*left,\*right;

};

int a[500000],cnt=5;

BST \*newnode(int item)

{

BST \*tmp=new BST;

tmp->data=item;

tmp->left=tmp->right=nullptr;

return tmp;

}

BST \*insert(BST \*root,int item)

{

if(root==0)return newnode(item);

else if(item<root->data)

root->left=insert(root->left,item);

else root->right=insert(root->right,item);

return root;

}

BST \*Find(BST \*root, int item) {

if(root == NULL) return NULL;

else if(root->data == item) return root;

else if(root->data < item) return Find(root->right,item);

else return Find(root->left,item);

}

BST \*findminimum(BST \*root)

{

while(root->left)

root=root->left;

return root;

}

BST \*findSuccessor(BST \*root,int item)

{

struct BST \*curr=Find(root,item);

if(curr==nullptr)return nullptr;

if(curr->right!=0) /// case 1:has right subtree

return findminimum(curr->right);

else

{ /// no right subtree

BST \*suc=0,\*ancestor=root;

while(ancestor!=curr)

{

if(curr->data>ancestor->data)

{

ancestor=ancestor->right;

}

else

{

suc=ancestor;

ancestor=ancestor->left;

}

}

return suc;

}

}

void print2DUtil(BST \*root, int space)

{

if (root == NULL)

return;

space += cnt;

print2DUtil(root->right, space);

cout<<endl;

for (int i = cnt; i < space; i++)

cout<<" ";

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

print2DUtil(root->left, space);

}

void print2D(BST \*root)

{

print2DUtil(root, 0);

}

int main()

{

int i,n,x;

printf("Enter the number of value to add: ");

cin>>n;

BST \*root=0;

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

{

cin>>a[i];

x=a[i];

root=insert(root,x);

}

print2D(root);

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

{

x=a[i];

BST \*suc= findSuccessor(root,x);

if(suc!=nullptr)

printf("The Successor of node %d is %d\n",x,suc->data);

else printf("The Successor of node %d doesn't exist\n",x);

}

}

/// heap sort

#include<bits/stdc++.h>

using namespace std;

int i,n,x,a[100000];

void heapify(int \*a ,int n,int i)

{

int largest=i;

int left=2\*i+1;

int right=2\*i+2;

if(left<n&&a[left]>a[largest])

largest=left;

if(right<n&&a[right]>a[largest])

largest=right;

if(largest!=i)

{

swap(a[largest],a[i]);

heapify(a,n,largest);

}

}

void heapsort(int \*a,int n)

{

for(i=n/2-1;i>=0;i--)

heapify(a,n,i);

for(i=n-1;i>=0;i--)

{

swap(a[0],a[i]);

heapify(a,i,0);

}

}

int main()

{

printf("Enter the number of elements to add: ");

cin>>n;

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

{

cin>>a[i];

}

heapsort(a,n);

cout<<"Sorted array : ";

for(i=0;i<n;i++)cout<<a[i]<<" ";

}