#include<bits/stdc++.h>

using namespace std;

class node{

public:

int data;

node\* right;

node\* left;

node(int d){

data = d;

right = NULL;

left = NULL;

}

};

node\* insertnode(){

int value;

cin>>value;

if(value==-1){

return NULL;

}

node \*root = new node(value);

root->left = insertnode();

root->right = insertnode();

return root;

}

void inorder(node \*root){

if(root==NULL){

return;

}

else{

inorder(root->left);

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

inorder(root->right);

}

}

void preorder(node\* root){

if(root==NULL){

return;

}

else{

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

preorder(root->left);

preorder(root->right);

}

}

void postorder(node\* root){

if(root==NULL){

return;

}

else{

postorder(root->left);

postorder(root->right);

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

}

}

void bfs(node\* root){

if(root==NULL){

return;

}

queue<node\*> q;

q.push(root);

while(!q.empty()){

node\* front = q.front();

cout<<front->data<<" ";

q.pop();

if(front->left!=NULL){

q.push(front->left);

}

if(front->right!=NULL){

q.push(front->right);

}

}

cout<<endl;

}

int height(node\* root){

if(root==NULL){

return 0;

}

int l = height(root->left);

int r = height(root->right);

int ans = 1 + max(l,r);

return ans;

}

int size(node \*root){

if(root==NULL){

return 0;

}

return 1+size(root->left)+size(root->right);

}

int sum(node\* root){

if(root==NULL){

return 0;

}

return root->data+sum(root->left)+sum(root->right);

}

int maximum(node \*root){

if(root==NULL){

return INT\_MIN;

}

int large = root->data;

int lr = max(maximum(root->left),maximum(root->right));

large = max(large,lr);

return large;

}

void topview(node \*root, int dist, int level, map<int,pair<int,int>> &mp){

if(root==NULL)

return;

if(mp.count(dist)==0){

mp[dist] = {root->data,level};

}

else{

if(mp[dist].second>level){

mp[dist] = {root->data,level};

}

}

topview(root->left,dist-1,level+1,mp);

topview(root->right,dist+1,level+1,mp);

}

void bottomView(node \*root, int dist, int level, map<int,pair<int,int>> &mpb){

if(root==NULL) return;

if(mpb.count(dist)==0){

mpb[dist] = {root->data,level};

}

else{

if(mpb[dist].second<level){

mpb[dist] = {root->data,level};

}

}

bottomView(root->left,dist-1,level+1,mpb);

bottomView(root->right,dist+1,level+1,mpb);

}

int main(){

//1 2 3 -1 4 -1 -1 -1 5 -1 -1

node\* root = insertnode();

map<int,pair<int,int>> mp;

topview(root,0,0,mp);

for(auto x: mp){

cout<<x.second.first<<" ";

}

cout<<endl;

map<int,pair<int,int>> mpb;

bottomView(root,0,0,mpb);

for(auto x: mpb){

cout<<x.second.first<<" ";

}

cout<<endl;

// inorder(root);

// cout<<endl;

// preorder(root);

// cout<<endl;

// postorder(root);

// cout<<endl;

// bfs(root);

// cout<<size(root)<<endl;

// cout<<sum(root)<<endl;

// cout<<maximum(root)<<endl;

// cout<<height(root)<<endl;

}