#include <cstdio>

#include <cstdlib>

#include <iostream>

#include <queue>

#define fi first

#define se second

using namespace std;

typedef pair<int,int> ii;

const int MAXN=1e5;

struct node{

int index,num;

struct node \*left,\*right;

}\*root;

int N;

int \*ar;

void update( node \*p ){

if( p->left && p->right )

p->num=1+p->left->num+p->right->num;

else if( p->left && !p->right )

p->num=1+p->left->num;

}

void add( node \*p , int k ){

if( !p->left ){

p->left=new node;

p->left->index=k;

p->left->num=1;

p->left->left=NULL;

p->left->right=NULL;

}

else if( !p->right ){

p->right=new node;

p->right->index=k;

p->right->num=1;

p->right->left=NULL;

p->right->right=NULL;

}

else if( p->left->num==p->right->num )

add(p->left,k);

else

add(p->right,k);

update(p);

}

void write(){

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

cout << ar[i] << ' ';

cout << endl;

}

void createTree(){

cout << "Enter N: ";

cin >> N;

root=new node;

ar=new int[N];

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

ar[i]=rand()%N;

root->index=ar[0];

root->num=1;

for( int i=1 ; i<N ; i++ )

add(root,ar[i]);

write();

}

ii inorderSearch( node \*p , int s ){

ii tmp=ii(0,0);

if( p->left ){

tmp=inorderSearch(p->left,s);

if( tmp.se )

return tmp;

}

cout << p->index << ' ';

if( p->index==s )

return ii(tmp.fi+1,1);

if( p->right ){

ii tmp2=inorderSearch(p->right,s);

tmp.fi+=tmp2.fi+1;

tmp.se+=tmp2.se;

return tmp;

}

return ii(1,0);

}

ii dfs( int s ){

int p=0,cnt=0;

node \*st[MAXN];

st[++p]=root;

while(p){

node \*nd=st[p--];

cnt++;

if( nd->left )

st[++p]=nd->left;

if( nd->right )

st[++p]=nd->right;

cout << nd->index << ' ';

if( nd->index==s ){

cout << endl;

return ii(cnt,1);

}

}

cout << endl;

return ii(cnt,0);

}

ii bfs( int s ){

int cnt=0;

queue<node \*> q;

q.push(root);

while( !q.empty() ){

node \*nd=q.front();

q.pop();

cnt++;

if( nd->left )

q.push(nd->left);

if( nd->right )

q.push(nd->right);

cout << nd->index << ' ';

if( nd->index==s ){

cout << endl;

return ii(cnt,1);

}

}

return ii(cnt,0);

}

int main(){

srand(time(NULL));

createTree();

int s;

while(22){

cout << "Please enter the number of the node that you want to search" << endl;

cin >> s;

if( s==-1 )

break;

cout << "Recursive Inorder:" << endl;

ii tmp=inorderSearch(root,s);

cout << endl;

if( tmp.se )

cout << "Node " << s << " has been found in " << tmp.fi << " steps" << endl;

else

cout << "Node " << s << " can not found in " << tmp.fi << " steps" << endl;

cout << endl << endl << endl << "BFS:" << endl;

tmp=bfs(s);

if( tmp.se )

cout << "Node " << s << " has been found in " << tmp.fi << " steps" << endl;

else

cout << "Node " << s << " can not found in " << tmp.fi << " steps" << endl;

cout << endl << endl << endl << "DFS:" << endl;

tmp=dfs(s);

if( tmp.se )

cout << "Node " << s << " has been found in " << tmp.fi << " steps" << endl;

else

cout << "Node " << s << " can not found in " << tmp.fi << " steps" << endl;

cout << endl << endl << endl;

getchar();

getchar();

system("clear");

write();

}

delete(ar);

return 0;

}