Bináris fa rekonstrukció

Adott egy n pontú bináris fa, amelynek pontjait az $1, \ldots, n$ számokkal azonosítjuk. Ismerjük a fa preorder és inorder bejárási sorrendjét. Rekonstruálni kell a fát.
A preorder sorrendet a következő algoritmus állítja elő.

```
Preorder(Fa, p)
    kiIr(Fa[p].adat)
    ha Fa[p].bal <> 0 akkor Preorder(Fa, Fa[p].bal)
    ha Fa[p].jobb <> 0 akkor Preorder(Fa, Fa[p].jobb)
eljárás vége
```

Az inorder sorrendet pedig az alábbi algoritmus adja.

```
Inorder(Fa, p)
    ha Fa[p].bal <> 0 akkor Inorder(Fa, Fa[p].bal)
    kiIr(Fa[p].adat)
    ha Fa[p].jobb <> 0 akkor Inorder(Fa, Fa[p].jobb)
eljárás vége
```

Feladat

Irjunk olyan programot, amely rekonstruálja a fát, tehát minden pontjára megadja, hogy annak ki a bal, és jobb fia!

Bemenet

A standard bemenet első sora egy egész számot tartalmaz, a fa pontjainak n ($1 \leq n \leq 100000$) számát. A második sor pontosan n pozitív egész számot tartalmat egy-egy szóközzel elválasztva, a fa preorder bejárási sorrendjét. A harmadik sor is sor pontosan n pozitív egész számot tartalmat egy-egy szóközzel elválasztva, a fa inorder bejárási sorrendjét.

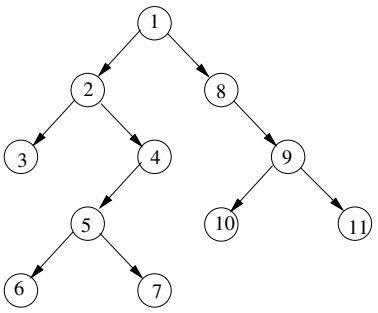
Kimenet

A standard kimenet pontosan n sort tartalmazzon, soronkét két egész számot. Az i-edik sor a fa i azonosítójú pontjának bal és jobb fiát. Hiányzó fiú esetén a 0 számot kell kiírni.

Példa

Bemenet

```
11
1 2 3 4 5 6 7 8 9 10 11
3 2 6 5 7 4 1 8 10 9 11
```



Kimenet

```
2 8
3 4
0 0
5 0
6 7
0 0
0 0
0 9
10 11
0 0
0 0
```

Korlátok

Időlimit: 0.1 mp.
Memórilimit: 32 MiB
Pontozás: a tesztesetek 40%-ában $n < 100$

Megoldás

Legyen

$$Preord[] = p_1, \ldots, p_n$$

a preorder,

$$Inord[] = i_1, \ldots, i_n$$

pedig az inorder sorrend. Jelölje $Hol[x]$ az x elen indexét az inorder sorrendben, azaz $Inord[Hol[x]] = x$. Ekkor a fa gyökere a p_1 pont. Legyen $k = Hol[p_1]$. Az $Inord[1..k-1]$ sorozat a bal részfa inorder bejárása, az $Inord[k+1..n]$ sorozat pedig a jobb részfa inorder bejárása. A $Preord[2..k]$ sorozat a bal részfa, a $Preord[k+1..n]$ pedig a jobb részfa preorder bejárása. Tehát ezek alapján a kiindulási problémát két részprobléma megoldására tudjuk bontani, mindkét részprobléma bibáris fa előállítását jelenti, tehát rekurzív megoldást kapunk.

```
1  #include <iostream>
2  #define maxN 100001
3
4  using namespace std;
5  typedef struct{
6     int bal,jobb;
7  } Par;
8  int Inord[maxN], Preord[maxN], Hol[maxN];
9  Par Fa[maxN];
10 int FaEpit(int tol1, int ig1, int tol2, int ig2){
11 //Globális: Preord[], Inord[], Hol
12    int x=Preord[tol1];
13    if(tol1==ig1){
14       Fa[x].bal=0; Fa[x].jobb=0;
15       return x;
16    }
17    int k=Hol[x];
18    int b=k-tol2;//a bal részfa elemeinek száma
19    if(b>0)
20       Fa[x].bal=FaEpit(tol1+1,tol1+b, tol2,k-1);
21    else
22       Fa[x].bal=0;
23    if(k+1<=ig2)
24       Fa[x].jobb=FaEpit(tol1+b+1,ig1,k+1,ig2);
25    else
26       Fa[x].jobb;
27    return x;
28 }
29 int main(){
30    int n;
31    cin>>n;
32    for(int i=1;i<=n;i++) cin>>Preord[i];
33    for(int i=1;i<=n;i++){
34       cin>>Inord[i];
35       Hol[Inord[i]]=i;
36    }
37    FaEpit(1,n,1,n);
38    for(int i=1;i<=n;i++)
```

```
39          cout<<Fa[i].bal<<"␣"<<Fa[i].jobb<<endl;
40      return 0;
41  }
```