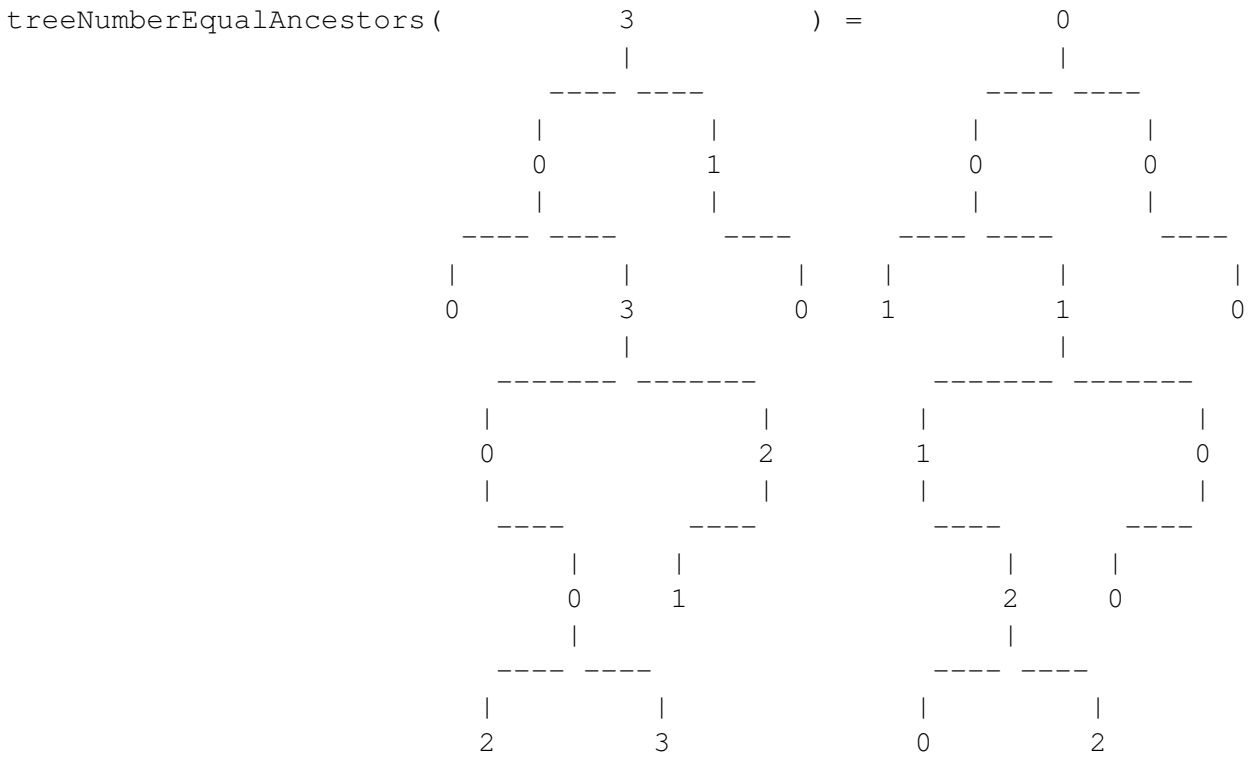
## Arbre del nombre de ancestres amb el mateix valor X59488_ca

Implementeu una funció **RECURSIVA** que, donat un arbre binari d'enters $t_1$, retorna un arbre binari d'enters $t_2$ amb la mateixa estructura que $t_1$ (mateix conjunt de posicions), i a on per a cada posició $p$, $t_2$ guarda a posició $p$ el nombre de nodes de $t_1$ en posicions ancestres de $p$ que guarden el mateix valor que el valor guardat a posició $p$.

```
// Pre:
// Post: Retorna un arbre d'enters t' amb la mateixa estructura que t,
//       i a on per a cada posició p, es cumpleix:
//       t'.p.value() = |{ p1 : t.p1.value() = t.p.value() i existeix p2 no bui
BinTree<int> treeNumberEqualAncestors(BinTree<int> t);
```

Aquí tenim un exemple de paràmetre d'entrada de la funció i la corresponent sortida:

```
treeNumberEqualAncestors( 3(0(0,3(0(,0(2,3)),2(1,))),1(,0)) )
                        = 0(0(1,1(1(,2(0,2)),0(0,))),0(,0))
```




Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc`, `BinaryTree.hh`, `treeNumberEqualAncestors.hh`. Us falta crear el fitxer `treeNumberEqualAncestors.cc` amb els corresponents `includes` i implementar-hi la funció anterior. Només cal que pugeu `treeNumberEqualAncestors.cc` al jutge.

### Entrada

La primera linia de l'entrada descriu el format en el que es descriuen els arbres, o bé INLINEFORMAT o bé VISUALFORMAT. Després venen un nombre arbitrari de casos. Cada

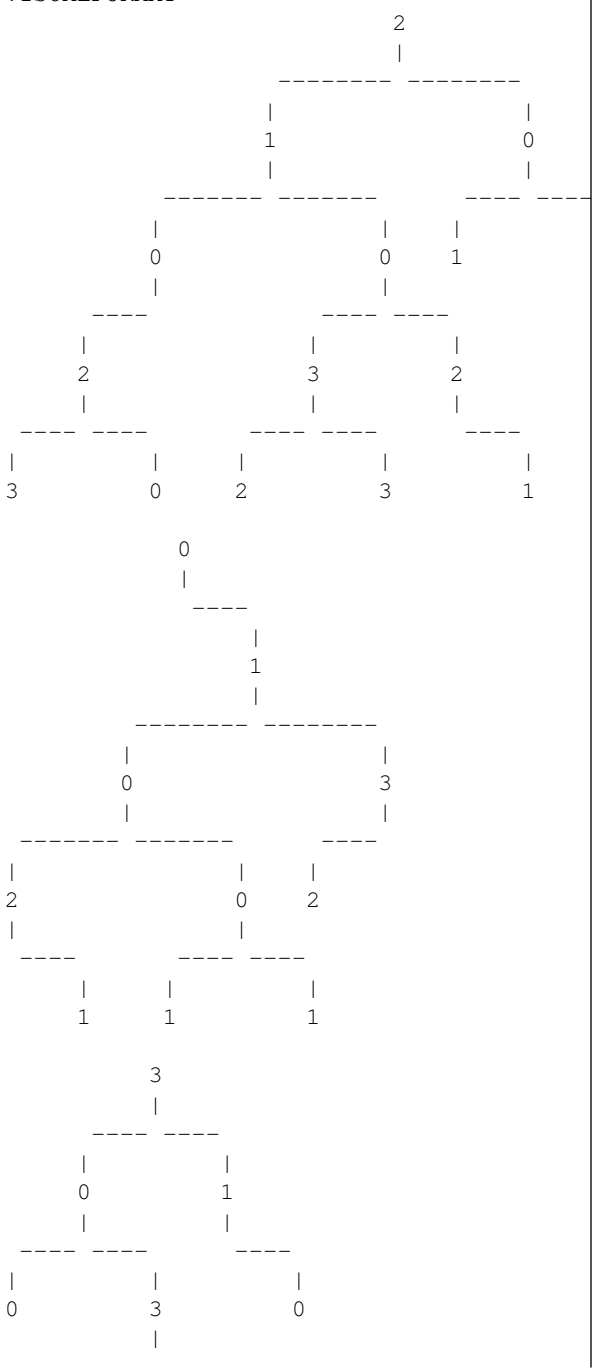
cas consisteix en una descripció d'un arbre un arbre binari d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

## Sortida

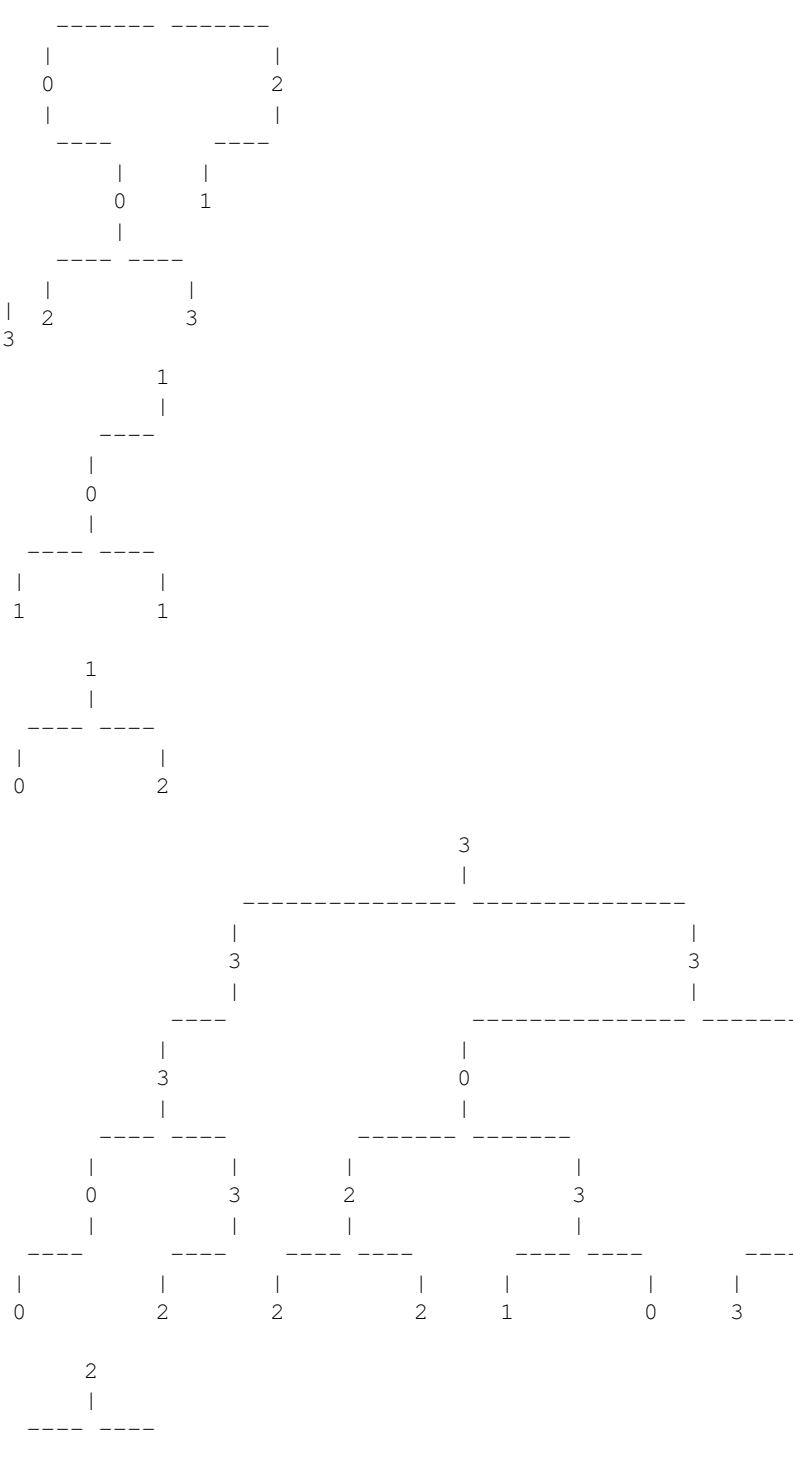
Per a cada cas, la sortida conté el corresponent resultat de la funció. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

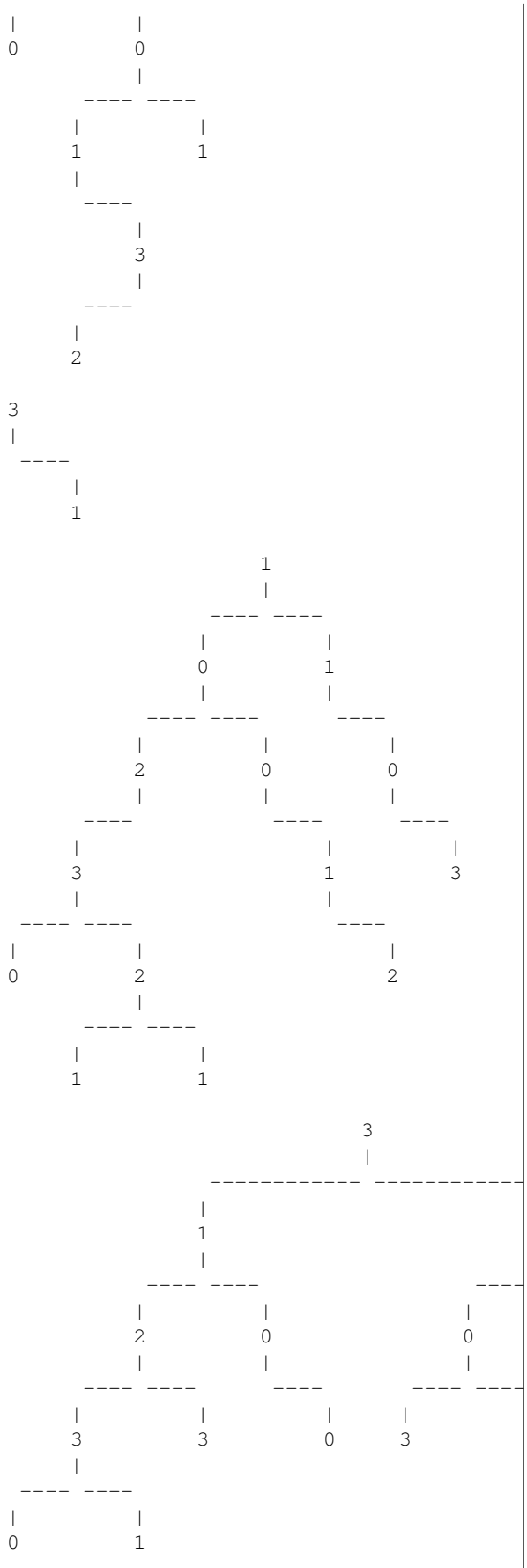
## Exemple d'entrada 1

```
VISUALFORMAT
```

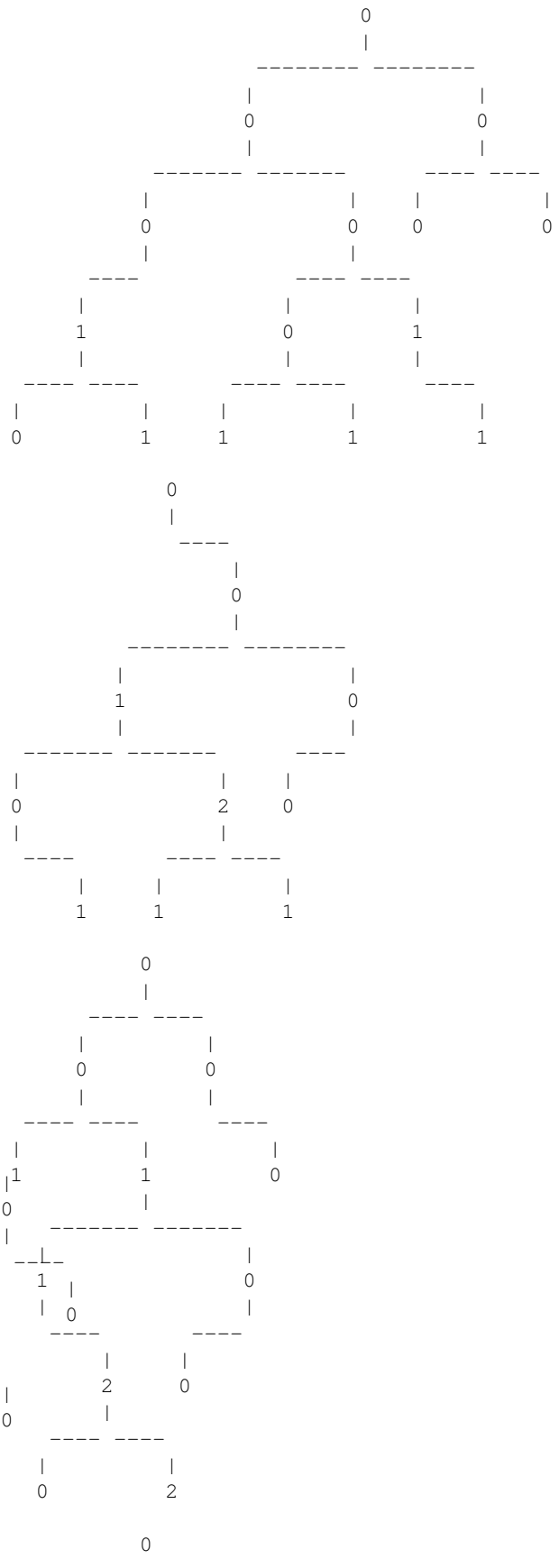


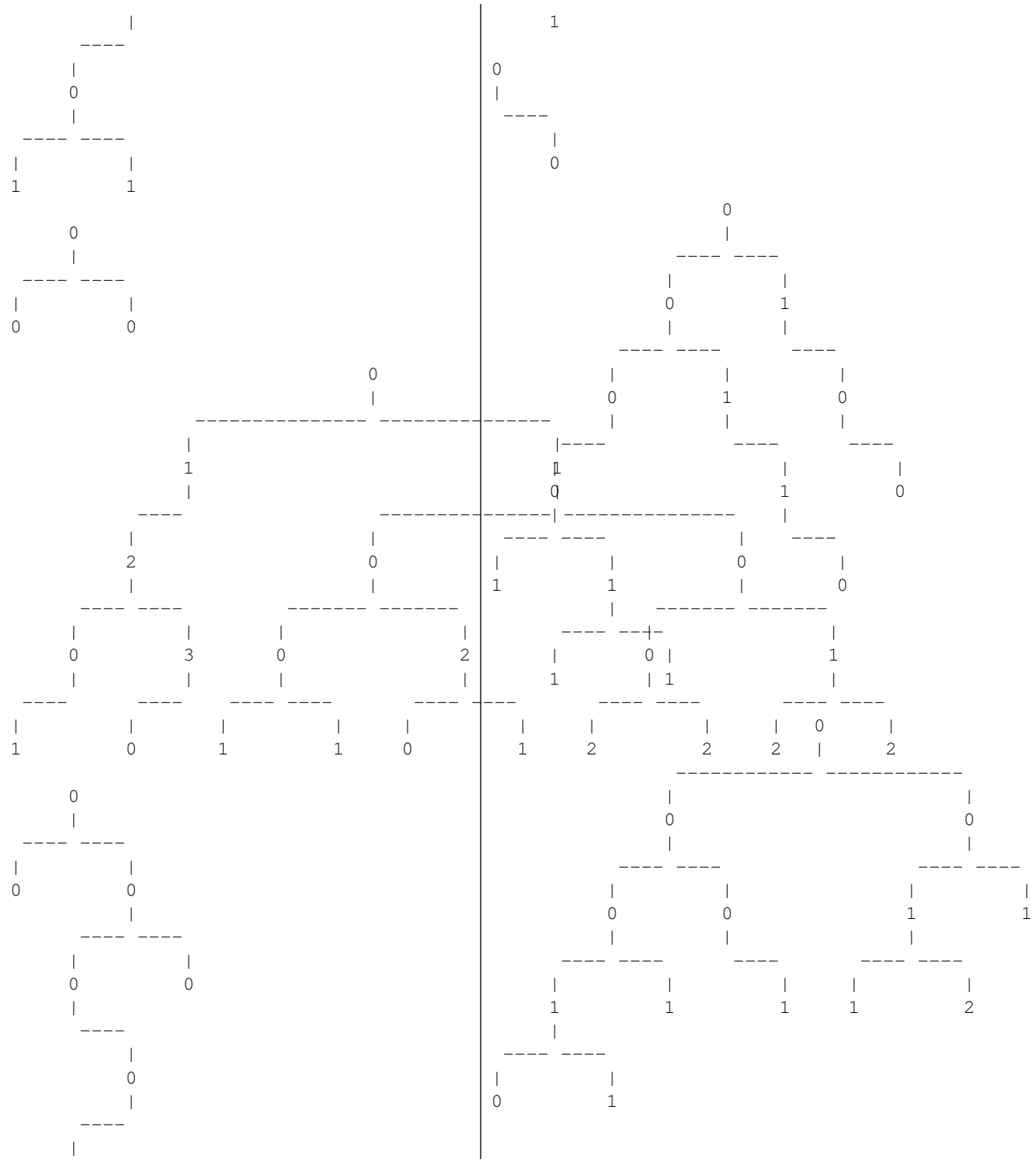






## Exemple de sortida 1





## Exemple d'entrada 2

```
INLINEFORMAT
2(1(0(2(3,0),),0(3(2,3),2(,1))),0(1,3))
0(,1(0(2(,1),0(1,1)),3(2,)))
3(0(0,3(0(,0(2,3)),2(1,))),1(,0))
1(0(1,1),)
1(0,2)
3(3(3(0(0,),3(2,)),),3(0(2(2,2),3(1,0)),0(2(3,3),0(0,0))))
2(0,0(1(,3(2,)),1))
3(,1)
```

```
1(0(2(3(0,2(1,1)),),0(,1(,2))),1(,0(,3)))
3(1(2(3(0,1),3),0(,0)),0(0(3,0),0))
```

### Exemple de sortida 2

```
0(0(0(1(0,1),),0(0(1,1),1(,1))),0(0,0))
0(,0(1(0(,1),2(1,1)),0(0,)))
0(0(1,1(1(,2(0,2)),0(0,))),0(,0))
0(0(1,1),)
```

```
0(0,0)
0(1(2(0(1,),3(0,)),),1(0(0(1,1),2(0,1)),0(0(2,2),1(2,2)
0(0,0(0(,0(1,)),0))
0(,0)
0(0(0(0(1,1(1,1)),),1(,1(,0))),1(,0(,0)))
0(0(0(1(0,1),1),0(,1)),0(1(1,2),1))
```

### Observació

Heu de trobar una solució **RECURSIVA** del problema. A més a més, la vostra funció i subfuncions que creeu han de treballar amb arbres. Però, si ho considereu oportú, podeu utilitzar memòria adicional de suport per mitjà d'alguna de les classes vistes durant el curs (**string, vector, stack, queue, list, map, set**).
Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, de cost $n \log(n)$ o menor, i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

### Informació del problema

Autoria: PRO2

Generació: 2026-01-25T17:10:57.717Z