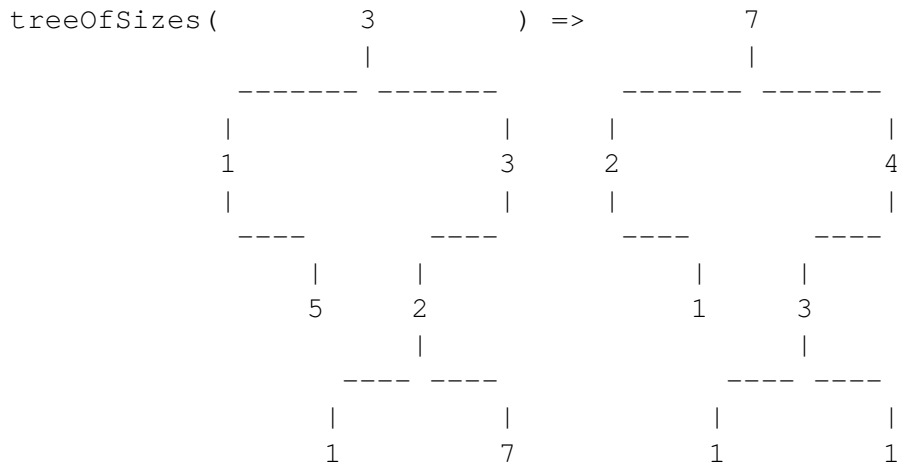


Arbre del nombre de descendents amb el mateix valor (copy)X98648_ca

Donat un arbre d'enters t , anomenem `numberEqualRoot(t)` al nombre de nodes de t que no son l'arrel de t però que guarden un valor igual al valor guardat a l'arrel de t . Per exemple: Implementeu una funció **RECURSIVA** que, donat un arbre binari d'enters t_1 , retorna un arbre d'enters t_2 amb la mateixa estructura que t_1 (mateix conjunt de posicions), i a on per a cada posició p , si t'_1 és el subarbre de t_1 a posició p i t'_2 és el subarbre de t_2 a posició p , llavors t'_2 a posició p guarda `numEqualRoot(t'_1)`. Aquesta és la capcelera:

```
// Pre:
// Post: Retorna un arbre d'enters t' amb la mateixa estructura que t,
//       i a on per a cada posició p, t'.p.value() = numberEqualRoot(t.p)
BinTree<int> treeNumberEqualRoot(BinTree<int> t);
```

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



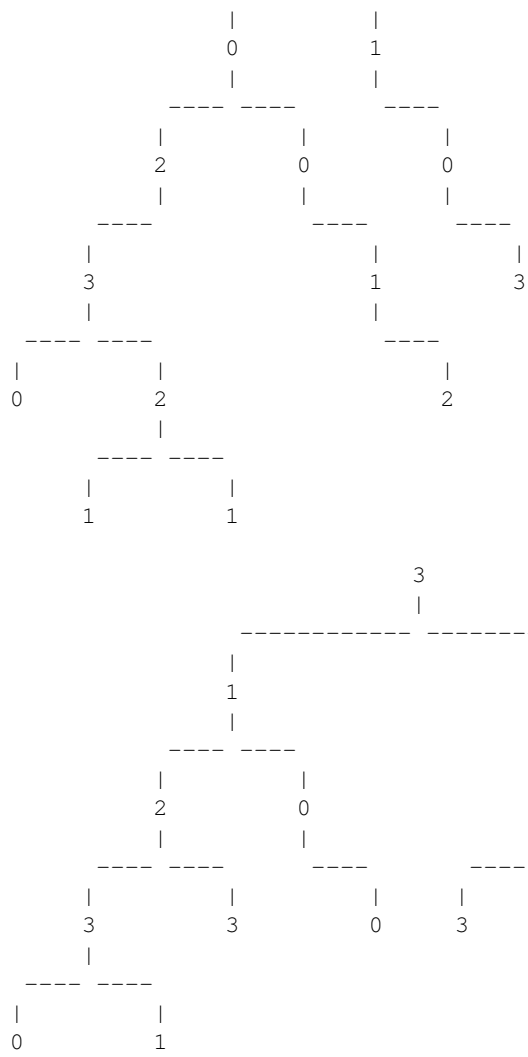
Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc`, `BinaryTree.hh`, `treeNumberEqualRoot.hh`. Us falta crear el fitxer `treeNumberEqualRoot.cc` amb els corresponents `includes` i implementar-hi la funció anterior. Només cal que pugeu `treeNumberEqualRoot.cc`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

Entrada

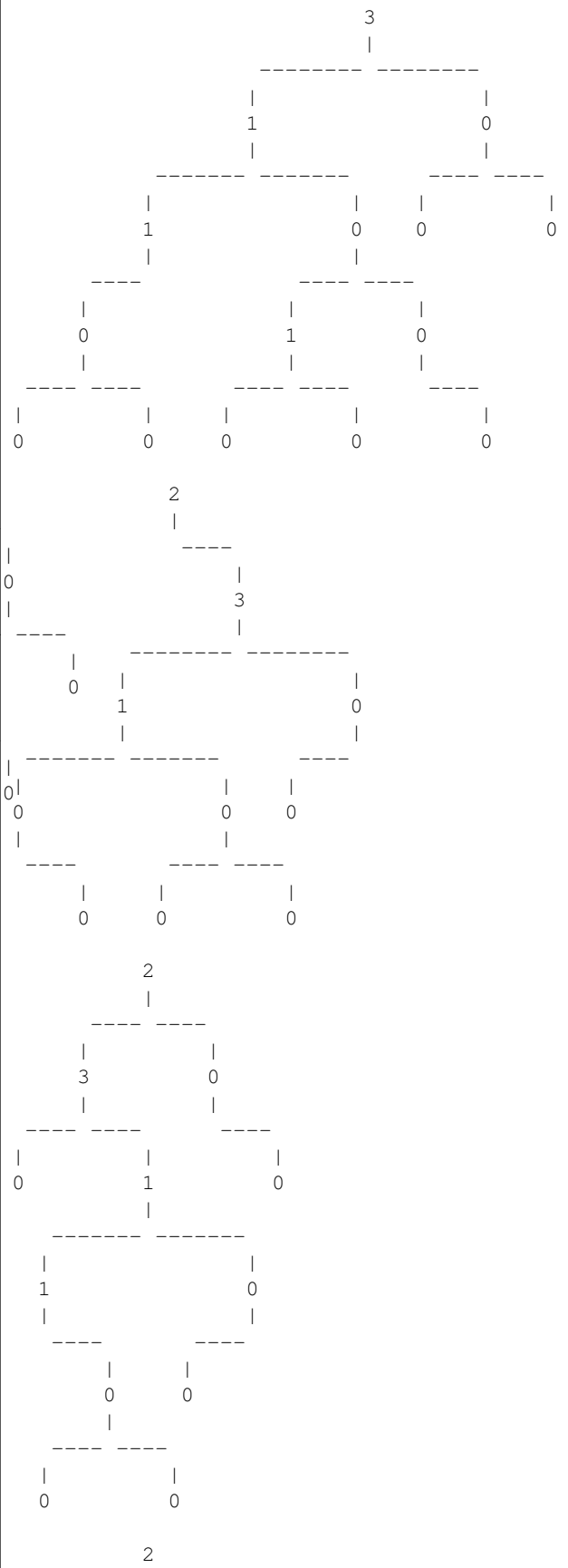
La primera línia de l'entrada descriu el format en el que es descriuen els arbres, o bé `IN-LINEFORMAT` o bé `VISUALFORMAT`. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció d'un arbre binari d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir 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 de sortida 1



Exemple de sortida 2

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

```
0(0,0)
7(2(1(1(0,),0(0,)),),3(1(2(0,0),0(0,0)),3(0(0,0),2(0,0))
1(0,0(0(,0(0,)),0))
0(,0)
4(2(1(0(0,0(0,0)),),0(,0(,0))),0(,0(,0)))
3(1(0(0(0,0),0),1(,0)),3(1(0,0),0))
```

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: 2026-01-25T17:40:28.478Z

© *Jutge.org*, 2006–2026.

<https://jutge.org>