

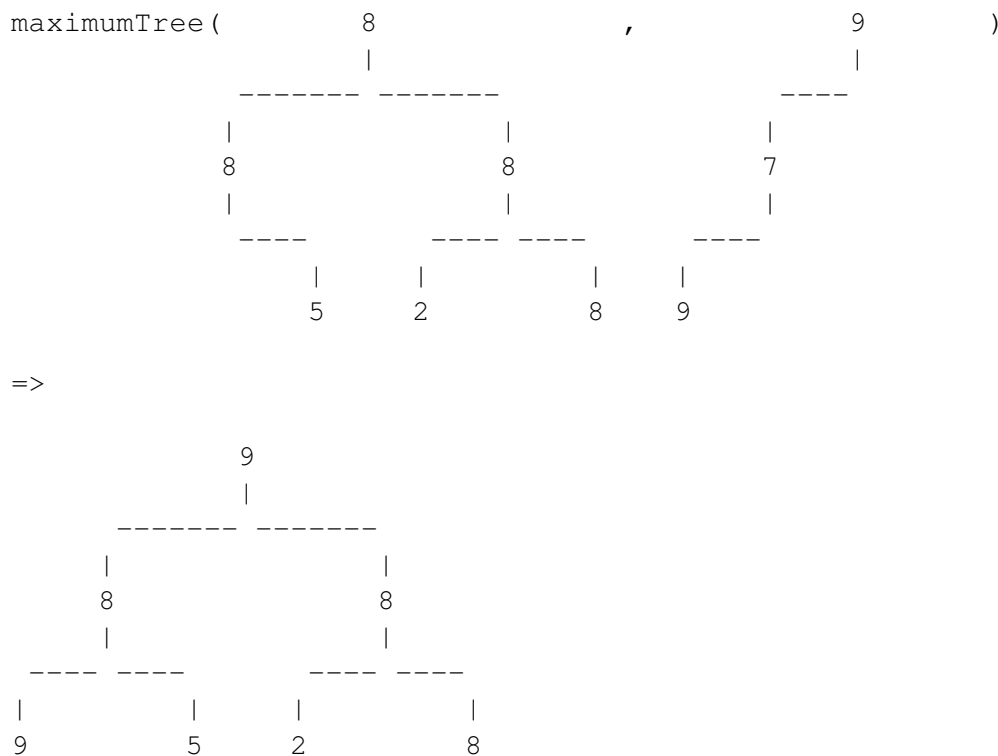
Arbre màxim

X47839_ca

Implementeu una funció **RECURSIVA** que, donats dos arbres binaris d'enters positius, obté un nou arbre que conté, per a cada posició, el màxim dels valors dels dos arbres de partida en les mateixes corresponents posicions. En cas que un dels arbres no tingui un valor definit en una posició, s'agafa el valor de l'altre arbre. Aquesta és la capelera:

```
// Pre: Rep dos arbres binaris d'enters positius t1 i t2.
// Post: Retorna un arbre, on a la seva arrel hi ha el màxim de les arrels de t1 i t2.
// en l'arrel del fill esquerre hi ha el màxim de les arrels dels fills esquerra i dreta de t1 i t2.
// en l'arrel del fill dret hi ha el màxim de les arrels dels fills drets de t1 i t2.
// i així successivament.
// Quan un dels arbres no té valors definits en alguna posició, l'arbre resulta el valor de l'altre arbre en aquella posició.
BinTree<int> maximumTree(BinTree<int> t1, BinTree<int> t2)
```

Aquí tenim un exemple d'entrada de la funció i la seva 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` `maximumTree.hh`. Només cal que creeu `maximumTree.cc`, posant-hi els includes que calguin i implementant la funció `maximumTree`. Només cal que pugueu `maximumTree.cc` al jutge.

Entrada

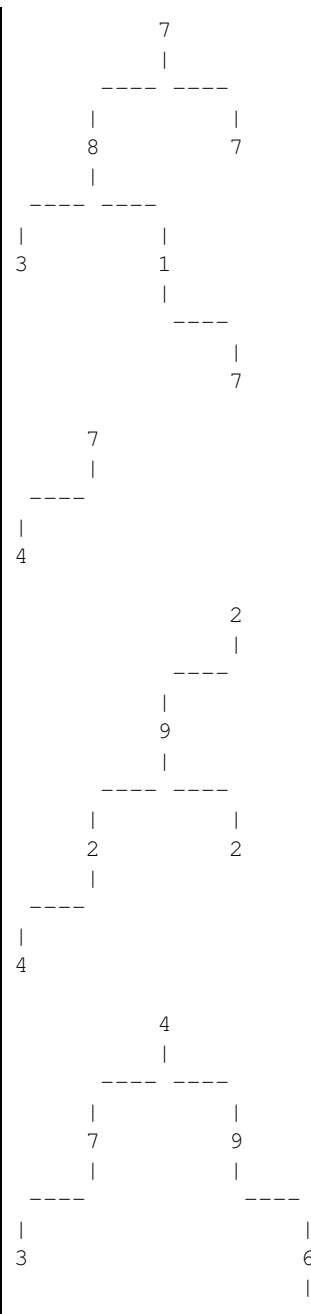
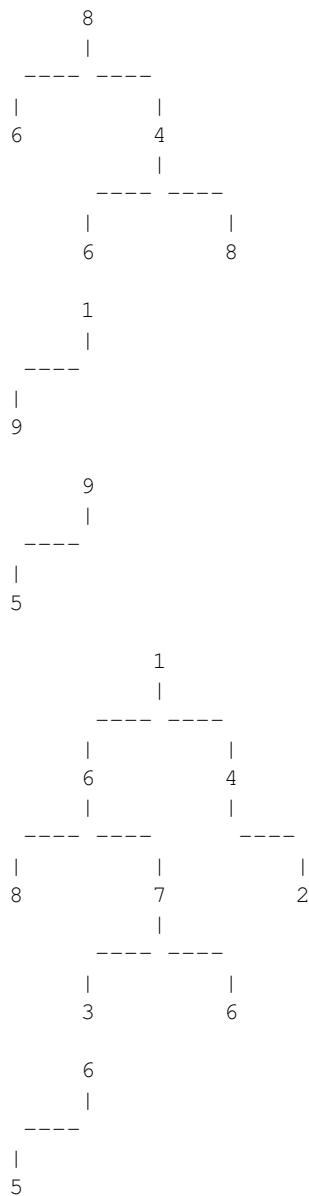
La primera línia de l'entrada descriu el format en el que es descriuen els arbres, o bé IN-LINEFORMAT o bé VISUALFORMAT. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció de dos arbres binaris d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

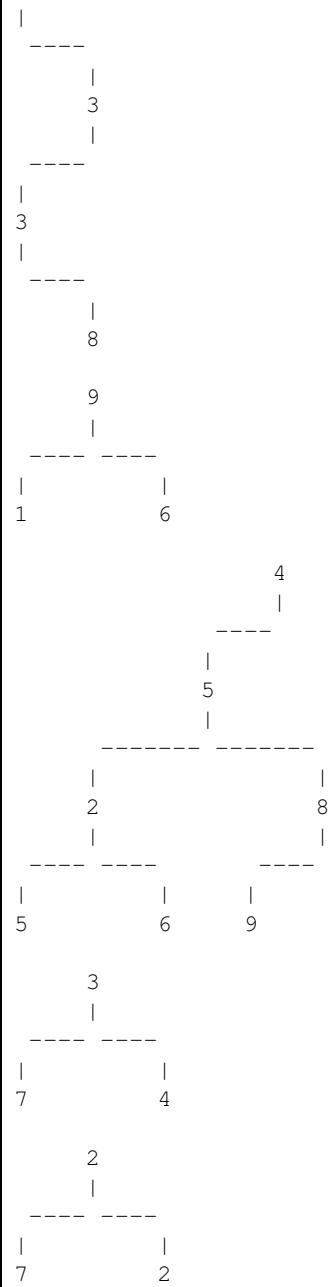
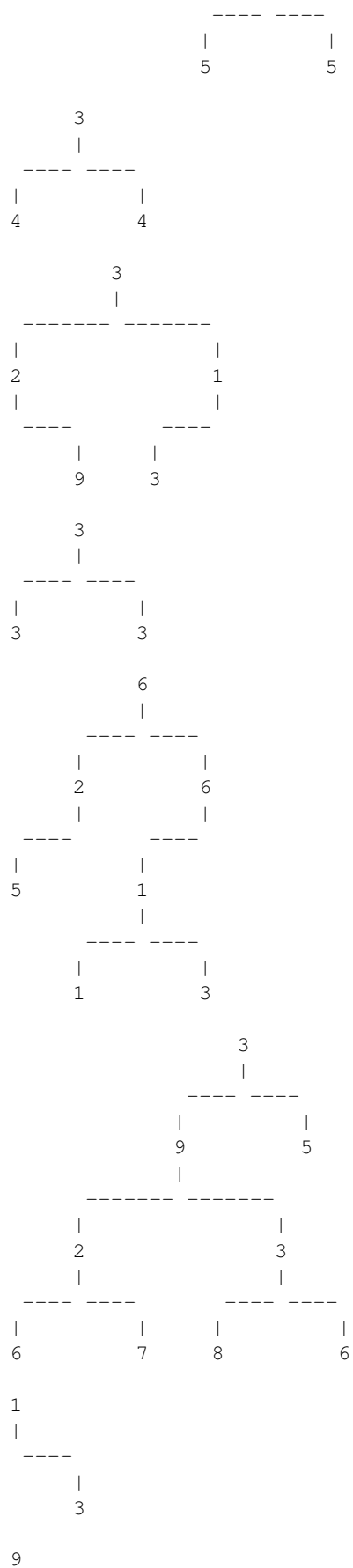
Sortida

Per a cada cas, cal escriure l'arbre binari resultant de calcular el màxim entre els dos arbres d'entrada. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

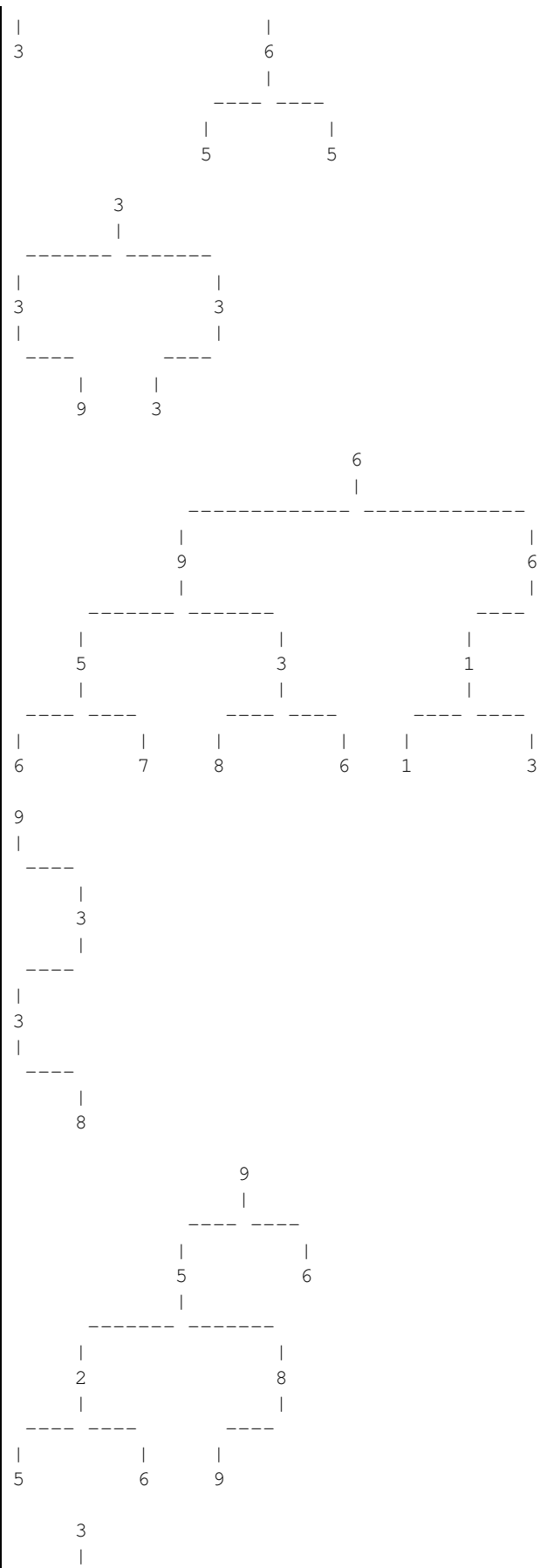
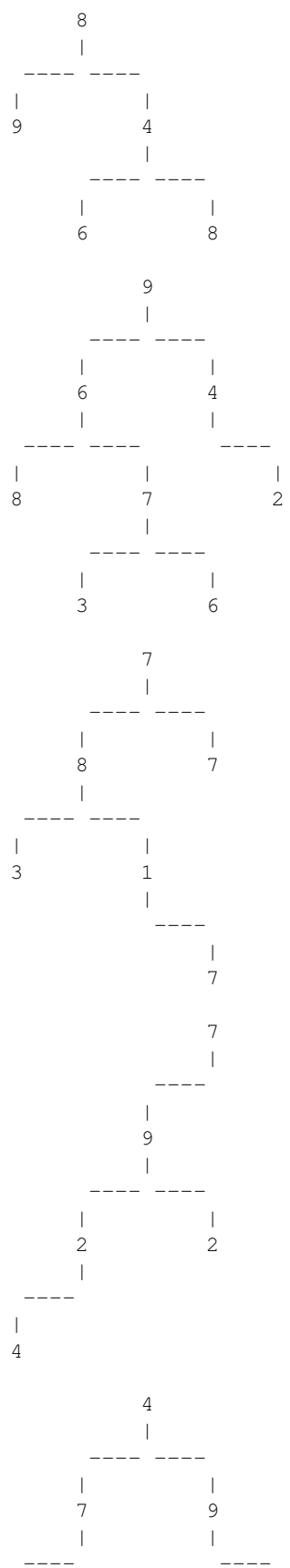
Exemple d'entrada 1

VISUALFORMAT





Exemple de sortida 1



| |

7 4

Exemple d'entrada 2

```
INLINEFORMAT
8(6,4(6,8))
1(9,)
9(5,)
1(6(8,7(3,6)),4(,2))
6(5,)
7(8(3,1(,7)),7)
7(4,)
2(9(2(4,),2),)
4(7(3,),9(,6(5,5)))
3(4,4)
3(2(,9),1(3,))
3(3,3)
6(2(5,),6(1(1,3),))
3(9(2(6,7),3(8,6)),5)
1(,3)
9(,3(3(,8),))
9(1,6)
4(5(2(5,6),8(9,)),)
3(7,4)
2(7,2)
```

Exemple de sortida 2

```
8(9,4(6,8))
9(6(8,7(3,6)),4(,2))
7(8(3,1(,7)),7)
7(9(2(4,),2),)
4(7(3,),9(,6(5,5)))
3(3(,9),3(3,))
6(9(5(6,7),3(8,6)),6(1(1,3),))
9(,3(3(,8),))
9(5(2(5,6),8(9,)),6)
3(7,4)
```

Observació

La vostra funció i subfuncions que creu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema.

Informació del problema

Autor : PRO2

Generació : 2023-10-21 13:48:28

© *Jutge.org*, 2006–2023.

<https://jutge.org>