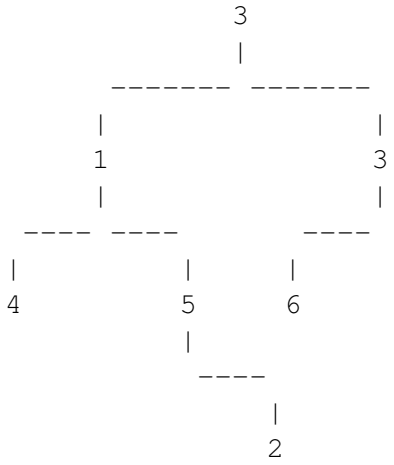
Camí descendent de suma màxima X65073_ca

`Preliminars` (**aquests preliminars son idèntics als de l'altre exercici d'arbres d'aquest mateix examen**)
Fixeu-vos en el següent arbre binari de naturals positius:



Com podeu observar, hi ha tres fulles amb valors 4, 2 i 6. Si agafem el camí descendent que ens porta des de l'arrel fins a la fulla amb un 4, i anem sumant tots els valors que trobem pel camí, obtenim 3+1+4 = 8. En canvi, si agafem en camí que ens porta des de l'arrel fins a la fulla amb un 2, la suma és 3+1+5+2 = 11. Pel cas de la fulla amb un 6, la suma obtinguda és 3+3+6 = 12. Per tant, 12 és la suma màxima que podem obtenir en un camí descendent des de l'arrel fins a alguna fulla.
`Fi de preliminars`
Implementeu una funció **RECURSIVA** que, donat un arbre binari d'enters, retorna la llista de valors que es troben des de l'arrel seguint el camí descendent fins a alguna fulla i que maximitza la suma d'aquests valors trobats. En cas que hi hagi varis camins màxims, s'haurà d'escollir el camí que va el més a l'esquerra possible. En el cas de l'exemple d'arbre anterior, el resultat de la funció seria [3,3,6]. Aquesta és la capcelera:

```
// Pre: t conté naturals positius en els seus nodes.
// Post: Retorna la llista de valors trobats en un camí descendent
//       des de l'arrel fins a alguna fulla. La fulla escollida fa que es maximi
//       la suma d'aquests valors trobats. En cas que hi hagi més d'un camí màx
//       s'escull el de més a l'esquerra possible.
list<int> descendingPathWithMaximumSum(BinTree<int> t);
```

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

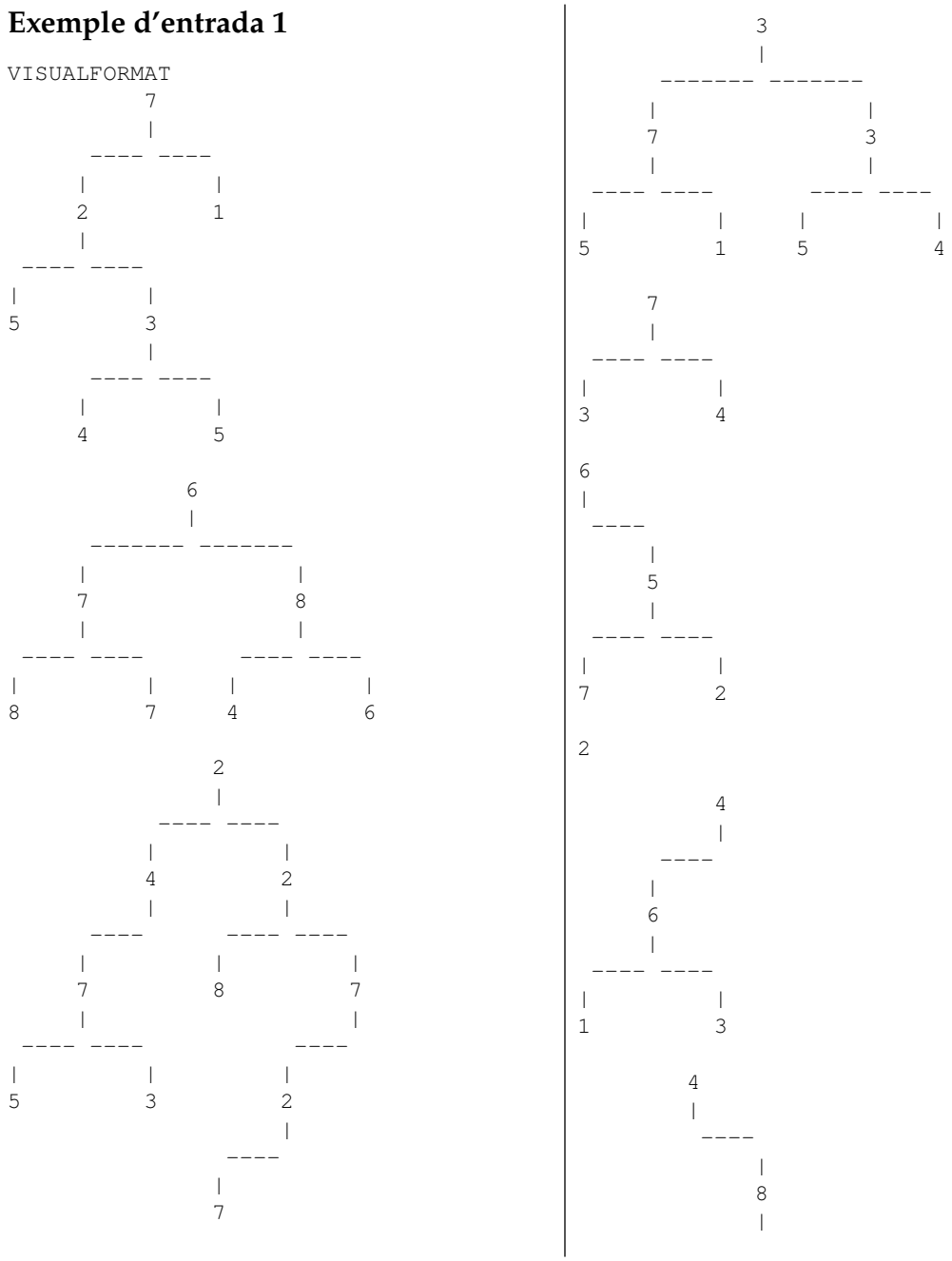
Entrada

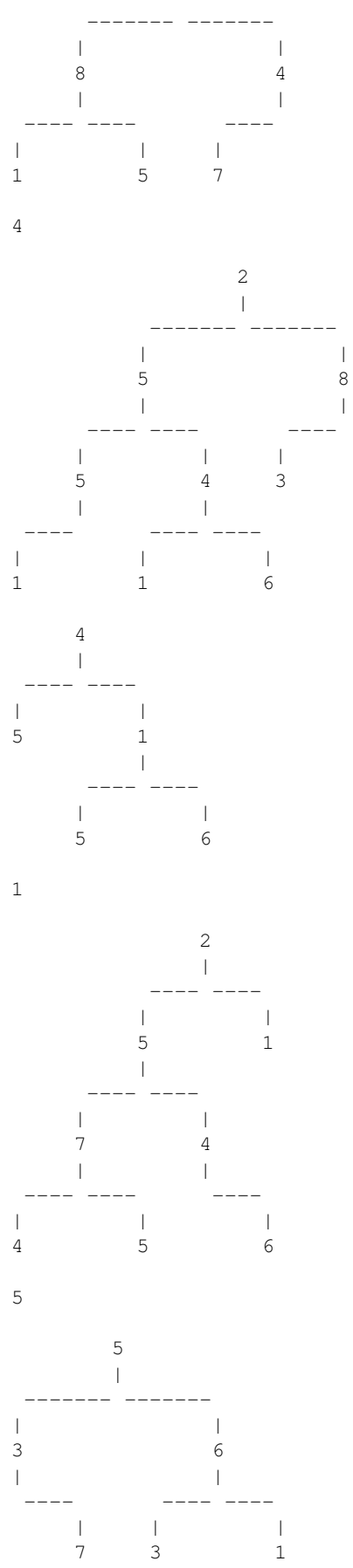
La primera linia de l'entrada descriu el format en el que es descriuen els arbres, o bé INLINEFORMAT o bé VISUALFORMAT. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció d'un arbre un arbre binari d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir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

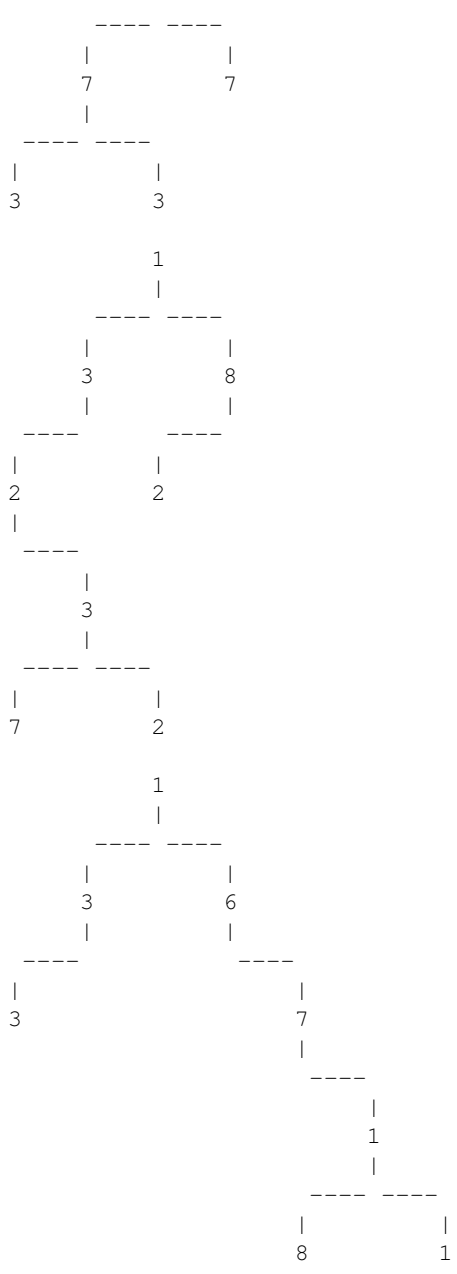
Sortida

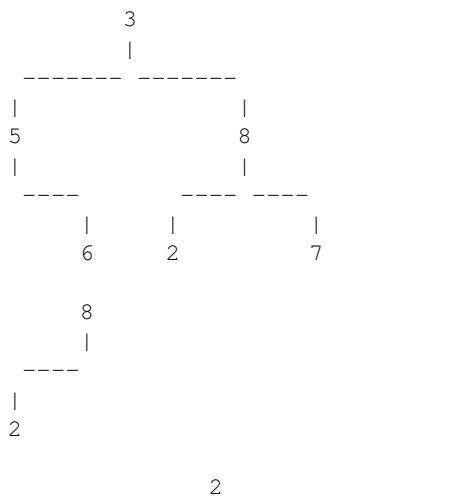
Per a cada cas, la sortida conté el corresponent resultat de cridar a la funció. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquest resultat. Només cal que implementeu la funció abans esmentada.

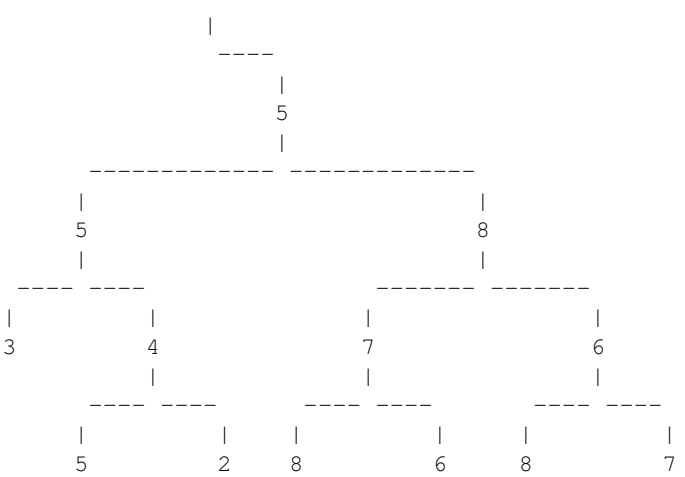
Exemple d'entrada 1

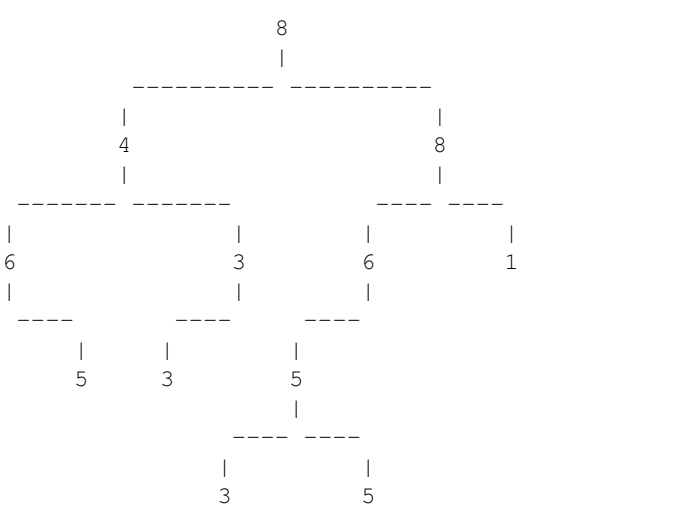


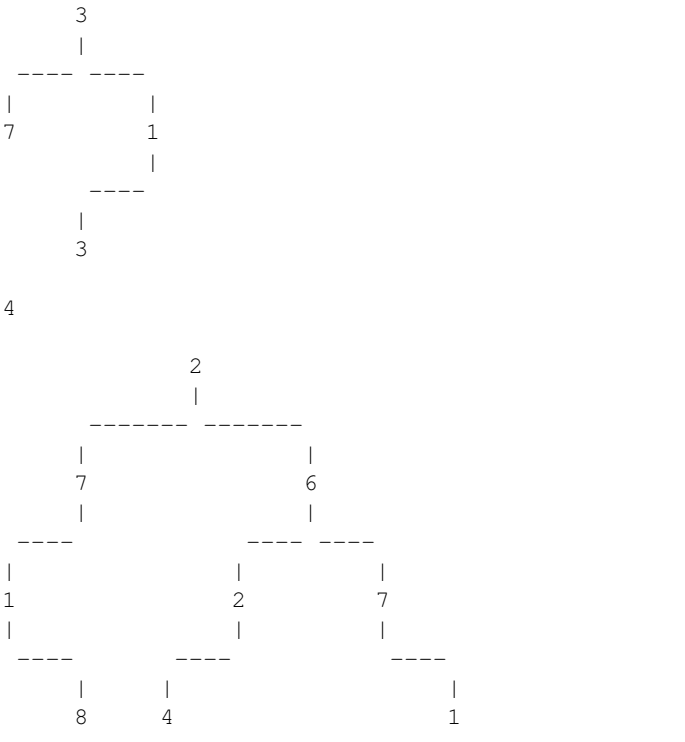


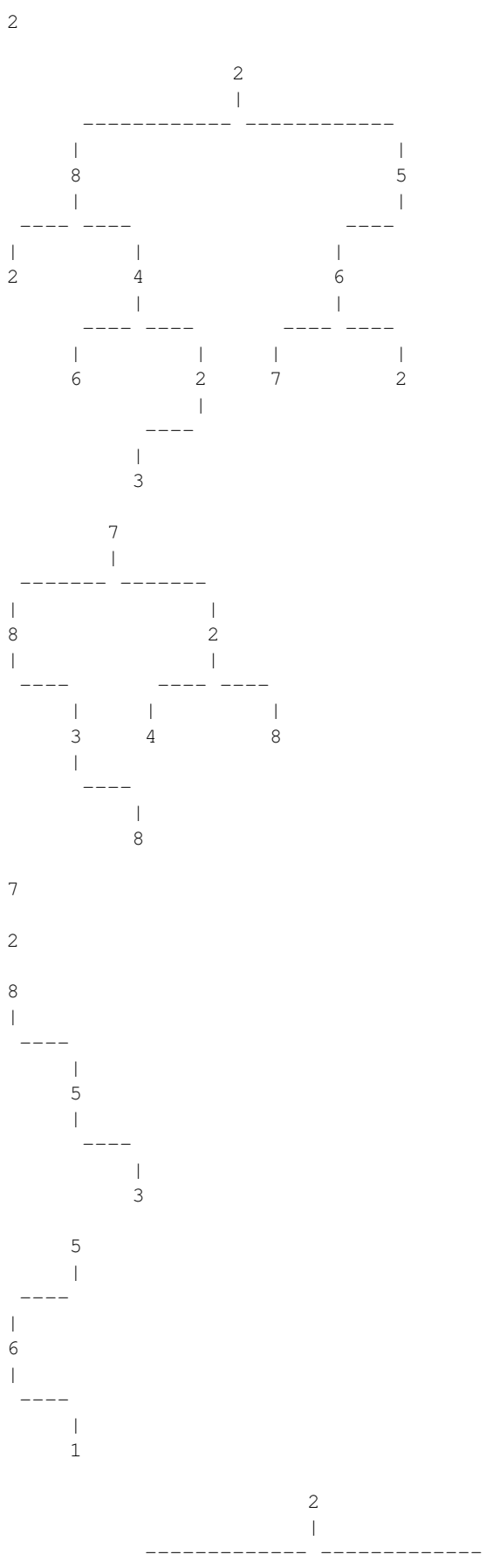


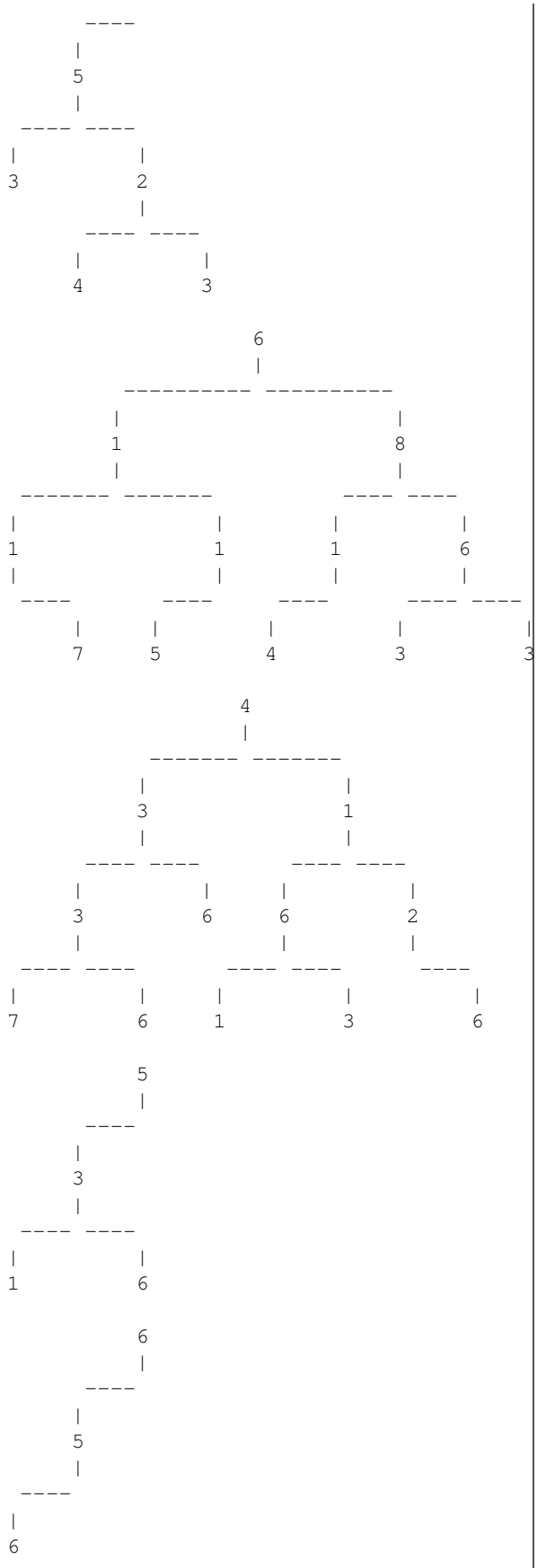












Exemple de sortida 1

```
[7,2,3,5]
[6,7,8]
[2,2,7,2,7]
[3,7,5]
[7,4]
[6,5,7]
[2]
[4,6,3]
[4,8,8,5]
[4]
[2,5,4,6]
[4,1,6]
[1]
[2,5,7,5]
[5]
[5,3,7]
[3,5]
[1,5,4,6,7]
[1,5,8]
[7,5,4]
[4]
[7,1,6]
[5]
[3,7,3]
[1,3,2,3,7]
[1,6,7,1,8]
[3,8,7]
[8,2]
[2,5,8,7,8]
[8,8,6,5,5]
[3,7]
[4]
[2,7,1,8]
[2]
[2,8,4,6]
[7,8,3,8]
[7]
[2]
[8,5,3]
[5,6,1]
[2,8,5,5,7]
[3]
[8,6,4,7]
[1,7,7,4,6]
[4,3,8]
[2,5,2,4]
[6,8,6,3]
[4,3,3,7]
[5,3,6]
[6,5,6]
```

Exemple d'entrada 2

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

Exemple de sortida 2

```
[7,2,3,5]
[6,7,8]
[2,2,7,2,7]
[3,7,5]
[7,4]
[6,5,7]
[2]
[4,6,3]
[4,8,8,5]
[4]
[2,5,4,6]
[4,1,6]
[1]
[2,5,7,5]
[5]
[5,3,7]
[3,5]
[1,5,4,6,7]
[1,5,8]
[7,5,4]
[4]
[7,1,6]
[5]
[3,7,3]
[1,3,2,3,7]
[1,6,7,1,8]
[3,8,7]
[8,2]
[2,5,8,7,8]
[8,8,6,5,5]
[3,7]
[4]
[2,7,1,8]
[2]
[2,8,4,6]
[7,8,3,8]
[7]
[2]
[8,5,3]
[5,6,1]
[2,8,5,5,7]
[3,4]
[8,6,4,7]
[1,7,7,4,6]
[4,3,8]
[2,5,2,4]
[6,8,6,3]
[4,3,3,7]
[5,3,6]
[6,5,6]
```

Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema. Avaluació sobre 10 punts:

- Solució lenta: 5 punts.

- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, de cost lineal 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: 2026-01-25T17:30:31.704Z