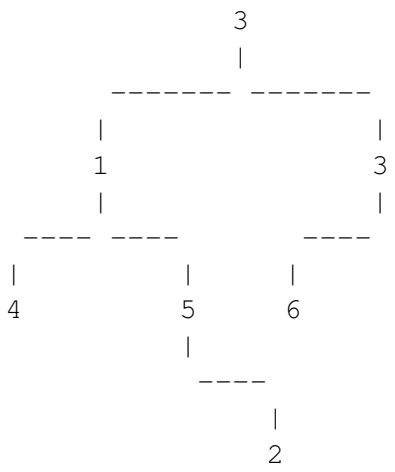
Màxima suma d'un camí descendent. X58333_es

Preliminares
Fijaos en el siguiente árbol binario de naturales positivos:



Como podéis observar, hay tres hojas con valores 4, 2 y 6. Si cogemos el camino descendente que nos lleva desde la raíz hasta la hoja con un 4, y vamos sumando todos los valores que encontramos por el camino, obtenemos 3+1+4 = 8. En cambio, si cogemos el camino que nos lleva desde la raíz hasta la hoja con un 2, la suma es 3+1+5+2 = 11. Para el caso de la hoja con un 6, la suma obtenida es 3+3+6 = 12. Por tanto, 12 es la suma máxima que podemos obtener en un camino descendente desde la raíz hasta alguna hoja.
Fin de preliminares
Implementad una función **RECURSIVA** que, dado un árbol binario de naturales positivos, devuelve la suma máxima que se puede obtener sumando los valores de los nodos que se encuentran en un camino descendente desde la raíz hasta alguna hoja. En el caso del ejemplo de árbol anterior, el resultado de la función sería 12. Esta es la cabecera:

```
// Pre: t contiene naturales positivos en sus nodos.
// Post: Retorna la suma máxima que se puede obtener sumando los valores que se
//       un camino descendente desde la raíz hasta alguna hoja.
int maxSumDescPath(BinTree<int> t);
```

Fijaos que el enunciado de este ejercicio ya ofrece unos ficheros que tendréis que utilizar para compilar: `main.cc, BinTree.hh, maxSumDescPath.hh`. Os falta crear el fichero `maxSumDescPath.cc` con los correspondientes `includes` e implementar la función anterior. Sólo hace falta que subáis `maxSumDescPath.cc` al juez.

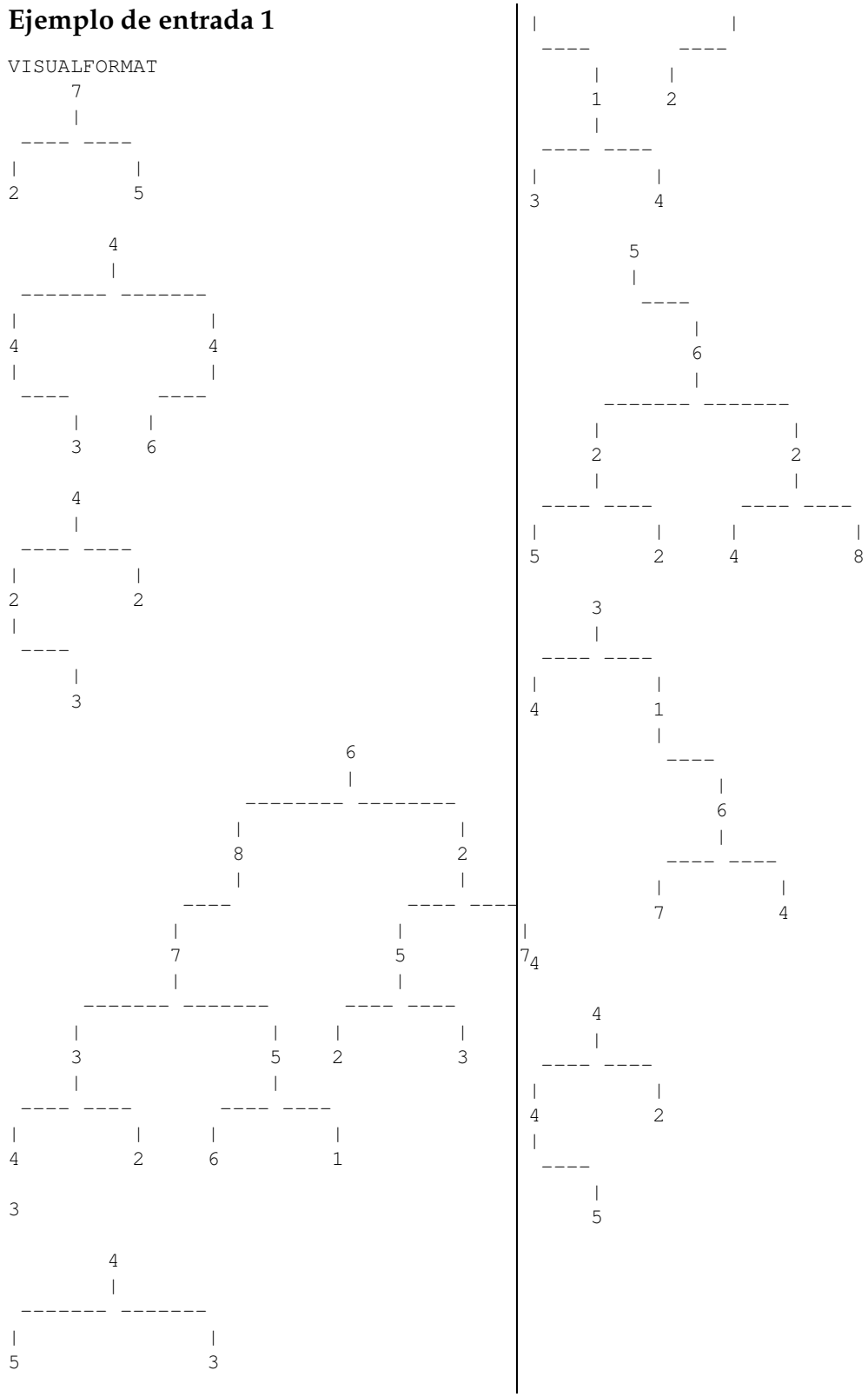
Entrada

La primera línea de la entrada describe el formato en el que se describen los árboles, o bien INLINEFORMAT o bien VISUALFORMAT. Después vienen un número arbitrario de casos. Cada caso consiste en una descripción de un árbol binario de enteros. Fijaos en que el programa que os ofrecemos ya se encarga de leer estas entradas. Sólo hace falta que implementéis la función antes mencionada.

Salida

Para cada caso, la salida contiene el correspondiente resultado de llamar a la función. Fijaos en que el programa que os ofrecemos ya se encarga de escribir este resultado. Sólo hace falta que implementéis la función antes mencionada.

Ejemplo de entrada 1



Ejemplo de salida 1

```
12
14
9
32
```

```
3
14
21
17
4
13
```

Ejemplo de entrada 2

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

Ejemplo de salida 2

```
12
14
9
32
3
14
21
17
4
13
21
31
14
13
24
16
24
3
14
```

Observación

Vuestra función y subfunciones que creéis deben trabajar únicamente con árboles. Debéis encontrar una solución **RECURSIVA** al problema. Evaluación sobre 10 puntos:

- Solución lenta: 5 puntos.
- Solución rápida: 10 puntos.

Entendemos como solución rápida una que es correcta, de coste lineal y capaz de superar los juegos de pruebas públicos y privados. Entendemos como solución lenta una que no es rápida, pero es correcta y capaz de superar los juegos de pruebas públicos.

Información del problema

Autor : PRO2
Generación : 2024-10-30 18:37:46