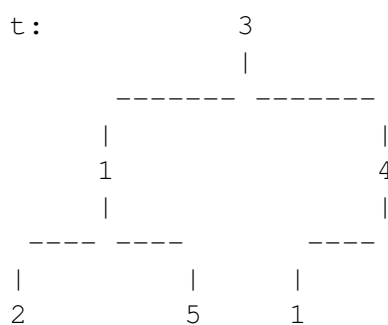

Màxim dels valors d'un arbre**X73911_ca**

Implementeu una funció **RECURSIVA** que, donat un arbre binari no buit d'enters, retorna el màxim dels seus valors. Aquesta és la capcelera:

```
// Pre: t és no buit
// Post: Retorna el màxim dels valors de t
int maxOfTree(BinaryTree<int> t);
```

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



=>

5

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `maxOfTree.hpp`. Us falta crear el fitxer `maxOfTree.cpp` amb els corresponents `includes` i implementar-hi la funció anterior. Quan pugueu la vostra solució al jutge, només cal que pugueu un tar construït així:

```
tar cf solution.tar maxOfTree.cpp
```

Entrada

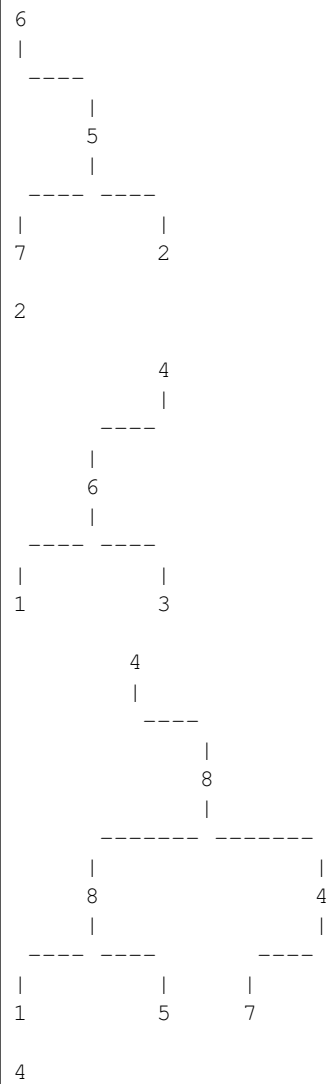
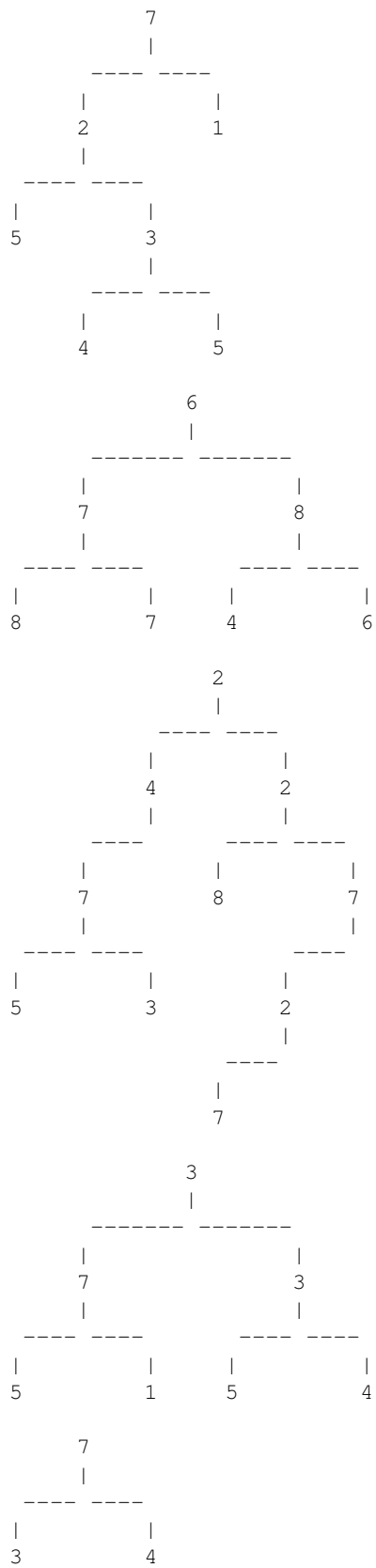
La primera línia de l'entrada descriu el format en el que es descriuen els arbres, o bé `IN-LINEFORMAT` o bé `VISUALFORMAT`. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció d'un arbre un arbre binari d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega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

Sortida

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

Exemple d'entrada 1

VISUALFORMAT



Exemple de sortida 1

7
8
8
7

7
7
2
6
8
4

Exemple d'entrada 2

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

Exemple de sortida 2

-2
-1
-1
-2
-2
-2
-7
-3
-1
-5

Exemple d'entrada 3

```
INLINEFORMAT
0 (55 (29, -47 (-15, 98)), -18)
-94 (82 (-21, 80), -16 (63, -85))
-27 (-50 (6 (13, -56), ), 23 (2, 36 (-2 (-37, ), )))
-56 (-5 (-100, -37), 7 (-70, -18))
5 (-3, -32)
50 (, -23 (-17, 91))
41
91 (59 (75, -46), )
55 (, 62 (-31 (-10, 69), -74 (67, )))
-56
12 (96 (-22 (88, ), 31 (15, -92)), -47 (70, ))
-58 (4, -1 (27, -35))
78
-91 (89 (35 (-95, -24), -50 (, 77)), -95)
-69
89 (-93 (, -72), -31 (-76, -91))
-25 (93, 76)
32 (-71, 73 (-68 (, -12 (, -70)), -86 (-61 (-68, 58), 93 (9)))
68 (-10 (22, 60), 91)
89 (-7 (-20, 37), )
```

Exemple de sortida 3

98
82
36
7
5
91
41
91
69
-56
96
27
78
89
-69
89
93
73
89

Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema. En les crides recursives, incloeu la hipòtesi d'inducció, és a dir una explicació del que es compleix després de la crida, i també la funció de fita/decreixement o una justificació de perquè la funció recursiva acaba.

Molt possiblement, una solució directa serà lenta, i necessitareu crear alguna funció recursiva auxiliar per a produir una solució més eficient capaç de superar tots els jocs de proves.

Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:24:46.937Z

© *Jutge.org*, 2006–2026.

<https://jutge.org>