Arbre d'alçades X22410_ca

Implementeu una funció **RECURSIVA** que, donat un arbre binari d'enters, retorna un nou arbre amb la mateixa estructura, i a on cada posició conté un número que és l'alçada del subarbre que penja d'aquella posició. Noteu que, si l'arbre és buit, llavors té alçada 0, i si l'arbre té un únic node (que serà arrel i fulla alhora), llavors té alçada 1. Aquesta és la capcelera:

```
// Pre:
// Post: Retorna un arbre d'enters amb la mateixa estructura que t,
//       i a on cada subarbre té com a arrel la seva alçada.
BinaryTree<int> treeOfHeights(BinaryTree<int> t);
```

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

```
treeOfHeights(3(1(,5),3(2(1,7),))) => 4(2(,1),3(2(1,1),))
```

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

```
tar cf solution.tar treeOfHeights.cpp
```

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en una línia amb un string describint 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 arbre d'alçades. 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

```
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
```

Exemple de sortida 1

```
4(3(1,2(1,1)),1)
3(2(1,1),2(1,1))
5(3(2(1,1),),4(1,3(2(1,),)))
3(2(1,1),2(1,1))
2(1,1)
3(,2(1,1))
1
3(2(1,1),)
4(,3(2(1,1),2(1,)))
1
```

Exemple d'entrada 2

```
0(55(29(-47(-15,98),),-18(86(-59(60(29(,-38
75(-46(-53(-48,-53),98(,61)),-49)
67(25,-50)
9(-87,25(95,))
15(-92(-47(70,),-87),)
4(-1(27,-35),)
78(86(-5(,68),),46(88(-59,-9(68,83)),79(89
-25(93(76(4,-8),-51(-22(-3,21),31(-34,32))
94(37(,6),72(-90(,24(,-38(55(-65,22),46)))
58
-20(82,81(-19,37))
97(-45(53(87(-96(-16(-35,97(,-23)),65(97,
-6(-10(,25(80,6(57,47))),-60(80,87))
40(-71(4(-17(90(,-4(,-57)),-67(,-87)),100
-14(-95(-31(41(-30(59(-71(27,-4),-75(,-92
8(54(11(-99(67(7,),),-47(-10,-18)),82(9,-
-69(-15(25(57(38(-54,-13),80),-5),39(,-5(
-53(19,35(9(29(-5,87),-60(21(-7,-16),)),
40(-49(-36,-47(51(-22(-7(-67(74(33,-100),
-9(-64(16,),49(-79,74))
```

Exemple de sortida 2

```
4(3(2(1,1),2(,1)),1)
2(1,1)
3(1,2(1,))
4(3(2(1,),1),)
3(2(1,1),)
3(2(1,),2(1,1)))
1
3(1,2(1,1))
5(4(,3(1,2(1,1))),2(1,1))
2(1,1))),2(1,))
3(2(1,),2(1,))
3(2(1,),2(1,1)))
```

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 cumpleix 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 produïr una solució més eficient capaç de superar tots els jocs de proves.

Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:03:46.999Z

© *Jutge.org*, 2006–2026.
https://jutge.org