Reemplaça els valors dels nodes a profunditat parell en un arbre per la suma per sota X20145_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 a profunditat parell conté la suma de nodes del subarbre que penja d'aquella mateixa posició a l'arbre inicial, i a cada posició a profunditat senar hi ha exactament el mateix valor que es troba en aquella posició a l'arbre inicial.
Sobreentenem que l'arrel de l'arbre està a profunditat 0, els nodes directes des de l'arrel són a profunditat 1, els nodes a distància dos de l'arrel són a profunditat 2, i així successivament.
Aquesta és la capcelera:

```
// Pre:  Sigui T el valor inicial de t.
// Post: Retorna un arbre d'enters R amb la mateixa estructura que T.
//       Per a cada posició p de T i R, si p és a profunditat senar,
//       llavors T i R tenen el mateix valor a posició p.
//       En canvi, si p es a profunditat parell, llavors el valor de R a posici
//       p és la suma de tots els valors que es troben a T a posició p o per so
BinaryTree<int> SumBelowAtEvenDepth(BinaryTree<int> t);
```

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

```
SumBelowAtEvenDepth(3(1(,5),3(2(1(3,6),7),))) => 31(1(,5),3(19(1(3,6),7),))
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `SumBelowAtEvenDepth.hpp`. Us falta crear el fitxer `SumBelowAtEvenDepth.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 SumBelowAtEvenDepth.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 de sumes. 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

```
1(2(6(,6),),8(0(,9(2,4)),8(1,0(,0))))
```

```
2(6(7(3(,5),1(,3)),),8)
0(9(,0(2,)),3(2,3(,0)))
2(0,1(,4(4,7)))
```

```
3(0(1(5(,6),5),0(6(,5),9(5,0))),0)
9(4,0(,0(,0)))
6(4(3(7(,2),3(,4)),0),5(4(,6(2,)),7(0(7,),0(0,))))
0(7(7(8,8),0(1,3)),)
0(0(,0(,3)),4(6,))
0(,5(,8(7(,0),0(0,0))))
5(8(3,),9(,0(2,9)))
9(6(5(5,),3(,0)),0)
0(0(9,4(0,3)),7)
1(3(,0(1,)),)
6(1(9(0(8,8),),4(0,8(,6))),8(4,3))
0(0(4,9),7(3(6,7),0))
3(0(2,8(0,)),1(6(0,8),8(8,4)))
1(4(1,3),2(8,6(2,7(3,))))
9(2,9(8(1(5,),1),2))
9(0,5(3(0,8),5))
```

Exemple de sortida 1

```
47(2(12(,6),),8(15(,9(2,4)),9(1,0(,0))))
35(6(19(3(,5),1(,3)),),8)
19(9(,2(2,)),3(2,3(,0)))
18(0,1(,15(4,7)))
45(0(17(5(,6),5),25(6(,5),9(5,0))),0)
13(4,0(,0(,0)))
60(4(19(7(,2),3(,4)),0),5(12(,6(2,)),14(0(7,),0(0,))))
34(7(23(8,8),4(1,3)),)
13(0(,3(,3)),4(6,))
20(,5(,15(7(,0),0(0,0))))
36(8(3,),9(,11(2,9)))
28(6(10(5,),3(,0)),0)
23(0(9,7(0,3)),7)
5(3(,1(1,)),)
65(1(25(0(8,8),),18(0,8(,6))),8(4,3))
36(0(4,9),7(16(6,7),0))
48(0(2,8(0,)),1(14(0,8),20(8,4)))
37(4(1,3),2(8,18(2,7(3,))))
37(2,9(15(1(5,),1),2))
30(0,5(11(0,8),5))
```

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.
Una solució directa superarà els jocs de proves públics i us permetrà obtenir una nota raonable. Però molt possiblement serà lenta, i necessitareu crear alguna funció recursiva auxiliar per a produïr una solució més eficient capaç de superar tots els jocs de proves.
Avaluació sobre 10 punts:

- Solució lenta: 6 punts.
- Solució lenta + justificació: 8 punts.
- solució ràpida: 8 punts.
- solució ràpida + justificació: 10 punts.

Entenem com a solució lenta una que és correcta i capaç de superar els jocs de proves públics.
Entenem com a solució ràpida una que és correcta i capaç de superar els jocs de proves públics i privats. La justificació val 1 punt i consisteix en definir correctament les PRE/POST de les funcions auxiliars que afegiu i en definir correctament les hipòtesis d'inducció i funcions de fita.

Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:02:55.933Z