Arbre màxim X77570_ca

Implementeu una funció **RECURSIVA** que, donats dos arbres binaris d'enters positius, obté un nou arbre que conté, per a cada posició, el màxim dels valors dels dos arbres de partida en les mateixes corresponents posicions. En cas que un dels arbres no tingui un valor definit en una posició, s'agafa el valor de l'altre arbre. Aquesta és la capcelera:

```
// Pre: Rep dos arbres binaris d'enters positius t1 i t2.
// Post: Retorna un arbre, on a la seva arrel hi ha el màxim de les arrels de t
// en l'arrel del fill esquerre hi ha el màxim de les arrels dels fills esquerre
// en l'arrel del fill dret hi ha el màxim de les arrels dels fills drets de t1,t
// i així successivament.
// Quan un dels arbres no té valors definits en alguna posició, l'arbre resultan
// el valor de l'altre arbre en aquella posició.

BinaryTree<int> maximumTree(BinaryTree<int> t1,BinaryTree<int> t2)
```

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

```
8(8(,5),8(2,8))
9(7(9,),)
=>
9(8(9,5),8(2,8))
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile, program.cpp, BinaryTree.hpp, maximumTree.hpp`. Només cal que creeu `maximumTree.cpp`, posant-hi els includes que calguin i implementant la funció `maximumTree`. I quan pugeu la vostra solució al jutge, només cal que pugeu un tar construït així:

```
tar cf solution.tar maximumTree.cpp
```

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en dues línies. Cadascuna d'aquestes dues línies té un string que descriu un arbre binari d'enters positius. 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, cal escriure l'arbre binari resultant de calcular el màxim entre els dos arbres d'entrada. 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

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

Exemple de sortida 1

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

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: 7 punts.
- Solució lenta + justificació: 8 punts.
- solució ràpida: 9 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: 2026-01-25T21:26:53.197Z