
Arbre de mides

X61092_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 el nombre total de nodes del subarbre que penja d'aquella mateixa posició a l'arbre inicial. 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 valor a l'arrel el nombre de nodes
//       del corresponent subarbre a t.
BinaryTree<int> treeOfSizes(BinaryTree<int> t);
```

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

```
treeOfSums(3(1(,5),3(2(1,7),))) => 7(2(,1),4(3(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`, `treeOfSizes.hpp`. Us falta crear el fitxer `treeOfSizes.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 treeOfSizes.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 mides. 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

```
7(5(1,3(1,1)),1)
7(3(1,1),3(1,1))
10(4(3(1,1),),5(1,3(2(1),)))
7(3(1,1),3(1,1))
3(1,1)
4(,3(1,1))
1
4(3(1,1),)
7(,6(3(1,1),2(1,)))
1
```

