Barreja d'arbres — X24235_ca

Implementeu una funció **RECURSIVA** que, donats tres arbres binaris amb la mateixa estructura, dos d'enters positius i un de caràcters d'operacions aritmètiques; obté un nou arbre amb la mateixa estructura que conté, per a cada posició, el resultat d'avaluar els valors dels nodes `t1` i `t2` amb l'operador del node `to`, tots tres en la mateixa posició. Aquesta és la capcelera:

```
// Pre: Rep dos arbres binaris d'enters positius {\tt t1} i {\tt t2} i un de ca
// Post: Retorna un arbre amb la mateixa estructura, on a cada node hi ha el re
BinaryTree<int> mixOfTrees(BinaryTree<int> t1,BinaryTree<int> t2, BinaryTree<ch
```

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

```
6(7(5,3),5(6,2))
9(1(2,7),0(9,3))
*(*(+,+),-(*,+))
=>
54(7(7,10),5(54,5))
```

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

```
tar cf solution.tar mixOfTrees.cpp
```

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en tres línies. En cadascun d'aquests, les dues primeres línies tenen un string que descriu un arbre binari d'enters positius i la tercera un arbre binari de caràcters. 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 retornar l'arbre binari resultant d'avaluar els valors dels nodes `t1` i `t2` amb l'operador del node `to`, tots tres en la mateixa posició. Tots els resultats han de ser nombres enters positius, es a dir, no poden haver nombres negatius. 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

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

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

Exemple de sortida 1

```
11(3(,12(2(7,0(,12)),7)),1(1(1,9),11(13,1)))
4(4,17(15(0,),9(6(4(,5),7(1,3)),2(5(12,24),48(4,7)))))
2(4(16,1),16(8,1(1(,12(9,)),25)))
8(,5(,1(3(,3),32(,2))))
4(,3(4(,2),15(3,9)))
8(42,63(,16))
18(6(64(,2),9(,1)),3(8(2,),16(9,18)))
1(24(7,1),2(3,0))
2(1(11,45),18(4,3))
2(18(7(48,),20(,32)),)
1(3(9(1,2(9,1)),5),)
0(1(7(16(8,),9),7(56(6,),7)),21(1,7))
4(11(6,),5(,1))
7(3(5,2(8(7,0),0(1,))),4(3,6))
4(0(0(5,),0(4,28)),1)
1(1(5(,12(2,0)),10(5(4,1),12)),15(6(9,54),))
3(18,3(2,))
11(,2(0(0,),))
0(,7(17(0,),2(49,5)))
2(9(5(4(7,1(1,9)),),35(6(5,2(0,3)),63(4(1,10),18(11,1)
```

Exemple d'entrada 2

```
6(3(2,2),)
0(6(3,9),)
-(+(+,*),)
```

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

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

Informació del problema

Autor : María Elizo i Xavi Vila
Generació : 2022-12-23 09:07:38

Exemple de sortida 2

```
6(9(5,18),)
2(10(8(,9),5),3(6,0))
2(2(6(7,),5(,3)),4(0,9(,30)))
6(4,0)
1(15,17)
15(,1(8,3(,8)))
1(1(56,15(13,11)),8)
6(3,35(,0(7,12)))
6(4,)
8(5(3,9),45(0,))
```