## Nombre de lletres a i b en un arbre binari — X38109_ca

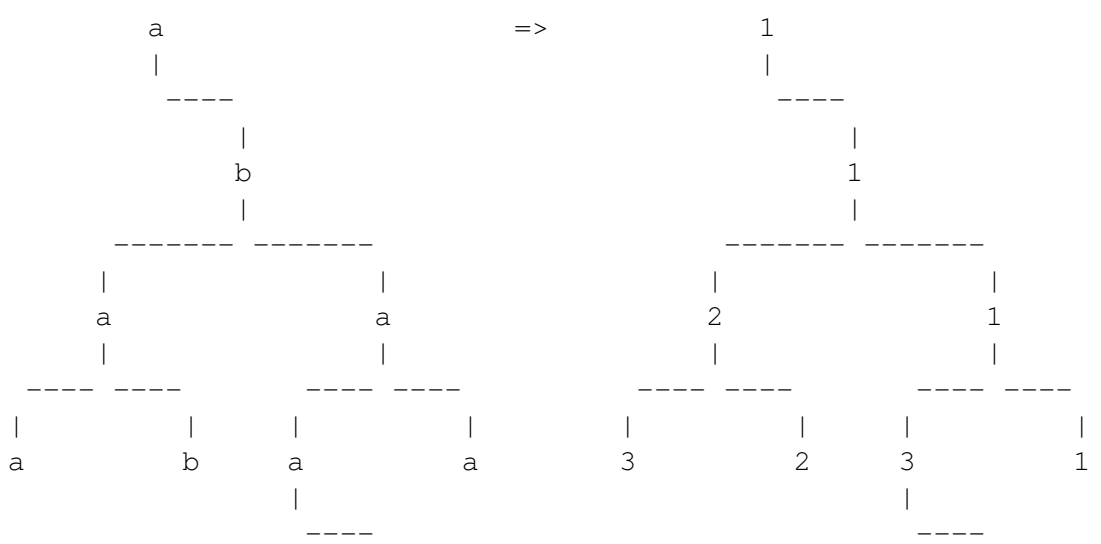
Implementeu una funció **RECURSIVA** que, donat un arbre binari d'strings amb valors `a` i `b` als nodes, retorna un nou arbre amb la mateixa estructura que l'inicial, i a on per a cada posició `p` hi ha un valor d'acord al següent criteri

1. Si el node és el fill esquerre del seu pare, ha de tenir el nombre de nodes `a` que hi ha des de l'arrel fins a aquell mateix node (ell inclòs).
2. Si el node és el fill dret del seu pare, ha de tenir el nombre de nodes `b` que hi ha des de l'arrel fins a aquell mateix node (ell inclòs).

```
/*
Pre:  T és un arbre binari que té com a valors els strings "a" o "b".
Post: Torna un arbre binari T' isomorf a T
      (T i T' tenen exactament la mateixa estructura).
      Per a tot node p de T', si p és fill esquerre del seu pare,
      llavors p té el nombre de nodes amb \verb#"a"# que hi ha des de l'arrel
      fins a la mateixa posició a l'arbre T.

      Per a tot node p de T', si p és fill dret del seu pare,
      llavors p té el nombre de nodes amb \verb#"b"# que hi ha des de l'arrel
      fins a la mateixa posició a l'arbre T.

      L'arrel de T' és sempre 1.
*/

BinaryTree<int> comptaAB(BinaryTree<string> t);
```

Aquí tenim un exemple de comportament de la funció:

```
comptaAB( a(,b(a(a,b),a(a(,b),a))) ) = 1(,1(2(2,2),1(3(,2),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, comptaAB.hpp`. Només cal que creeu `comptaAB.cpp`, posant-hi els includes que calguin i implementant la funció `numLeftRight`. I quan pugeu la vostra solució al jutge, només cal que pugeu un tar construït així:

```
tar cf solution.tar comptaAB.cpp
```

## Entrada

La primera linia de l'entrada descriu el format en el que es descriuen els arbres, o bé INLINEFORMAT o bé VISUALFORMAT. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció d'un arbre un arbre binari d'enters amb valors -1 i -2. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

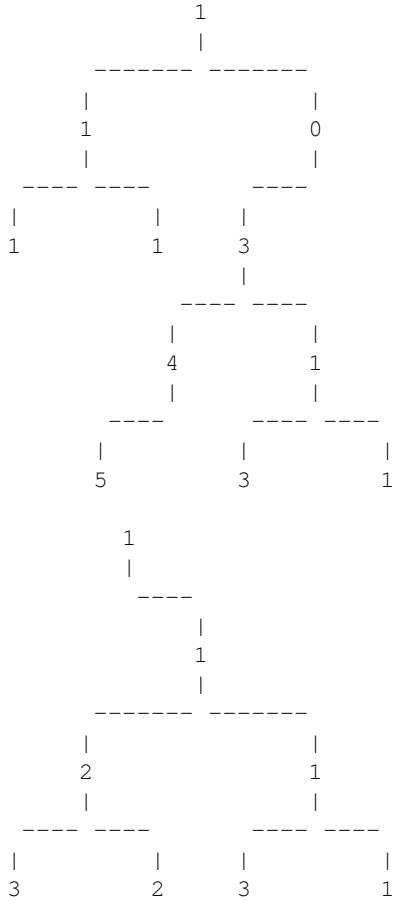
## Sortida

Per a cada cas, cal escriure l'arbre binari resultant de cridar a la funció abans esmentada amb l'arbre 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

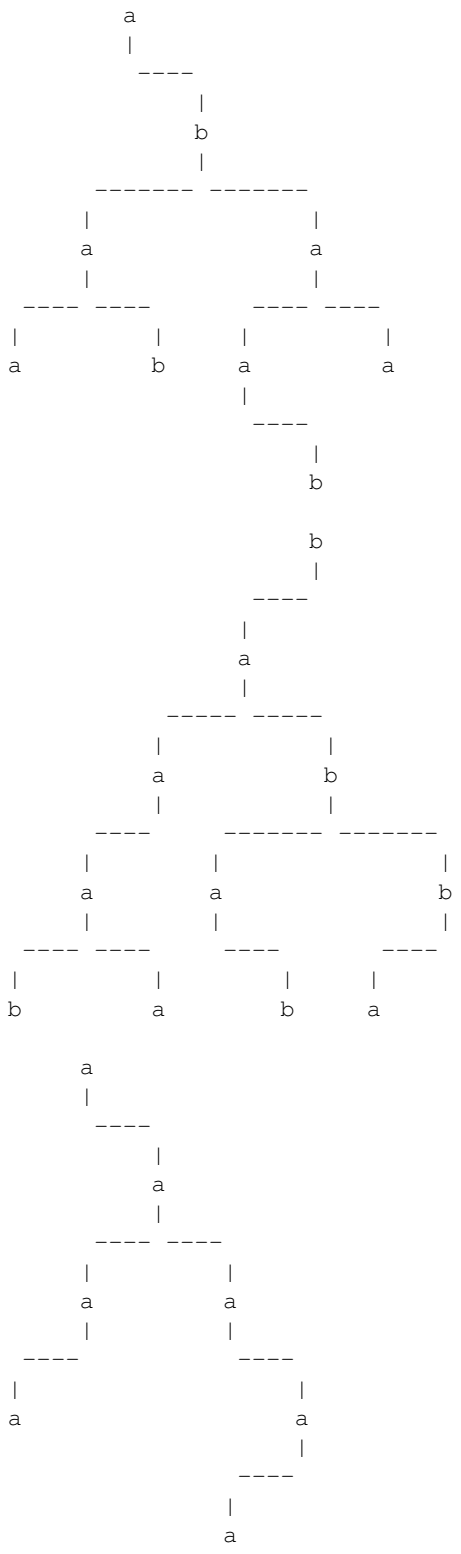
```
INLINEFORMAT
a(b(b,a),a(a(a(a,),b(b,a)),))
a(,b(a(a,b),a(a(,b),a)))
```

## Exemple de sortida 1



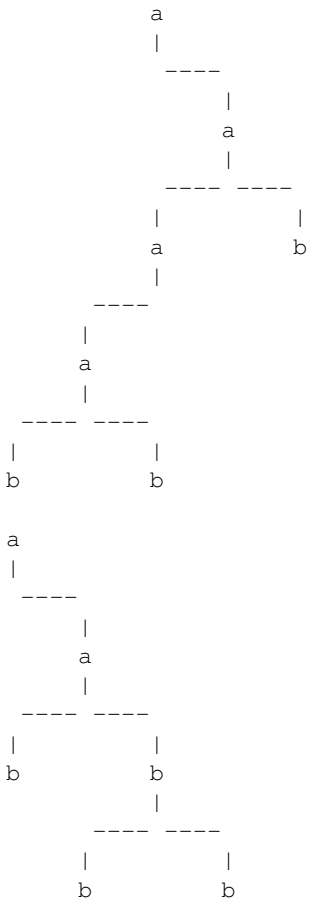
## Exemple d'entrada 2

```
VISUALFORMAT
```

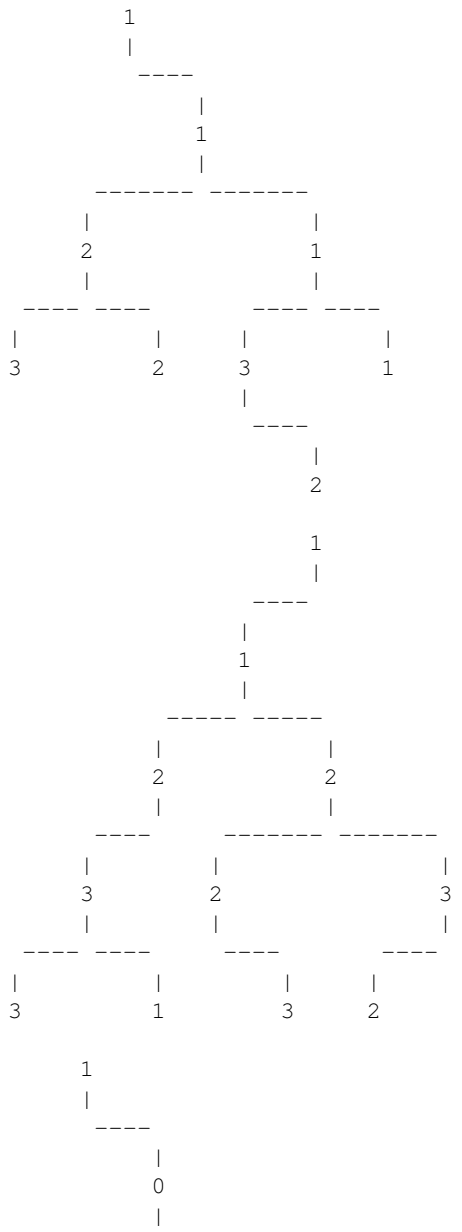


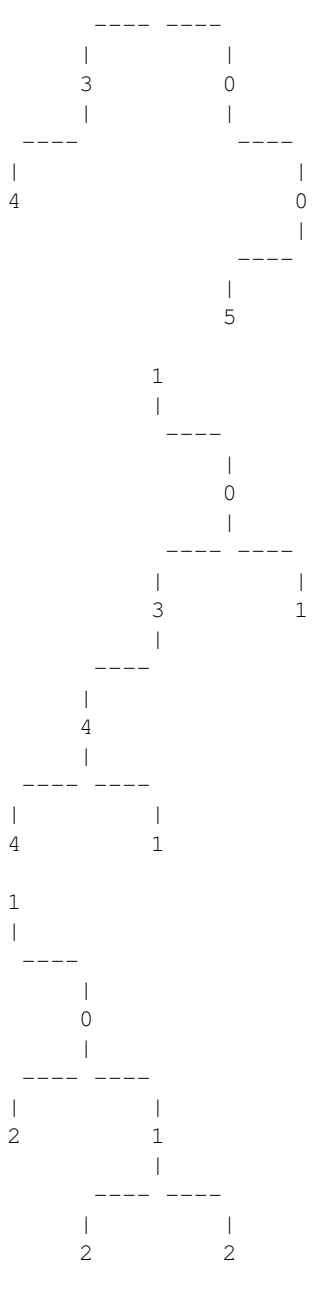


2



**Exemple de sortida 2**





## 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 funció de fita/decreixement.
Avaluació sobre 10 punts:

- Jocs de prova públics, semàfor verd: 6 punts.
- Jocs de prova privats, semàfor verd: 4 punts

Coses que poden restar punts a la puntuació anterior:

- Recursiu en comptes d'iteratiu (o viceversa): zero de nota final.

- Utilització d'estructures de dades auxiliars diferents del tipus que apareix a l'enunciat: des de -5 fins a zero de nota final. En cas de dubte, demaneu al professor.
- No fer immersió de paràmetres/resultats si això millora el cost assimptòtic del vostre codi: de -5 cap endavant, depenent de la gravetat.
- Manipulació *excessiva* d'estructures de dades: de -5 cap endavant. Exemple: agafar una pila i capgirar-la més del que cal.
- No posar PRE/POST a les funcions auxiliars que feu servir: -3. Òbviament, ja us donem la PRE/POST de la funció que us demanem, no cal que la repetiu. Ara bé, si feu una funció que implementa una generalització de la funció que us demanem, sí que cal que en feu la PRE/POST.
- No posar l'invariant si feu un bucle: -2.

### Informació del problema

Autoria: PRO1

Generació: 2026-01-25T21:10:32.293Z