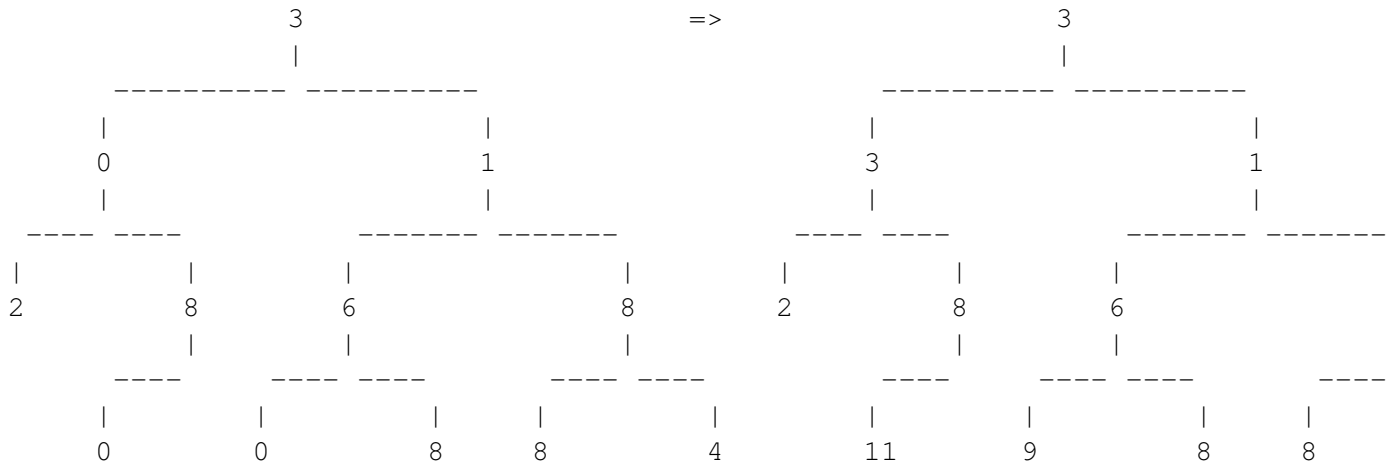
## (BinaryTrees) Reemplaça 0s per suma per sobre a profunditat parell X84631_ca

Implementeu una funció **RECURSIVA** que, donat un arbre binari de naturals, retorna un nou arbre que és idèntic a l'inicial, excepte que cada 0 s'ha reemplaçat per la suma dels elements a profunditat parell que apareixen per sobre d'aquest 0 (en l'arbre original), és a dir, les posicións a profunditat parell que són antecessores d'aquest 0.
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 l'arbre t que es rep com a paràmetre.
//       Els valors guardats a T son majors o iguals a 0.
// Post: Sigui T' l'arbre retornat. T i T' tenen exactament la mateixa estructu
//       A més a més, per a cada posició p de T', si T té un valor x diferent d
//       llavors T' també té x a posició p.
//       En canvi, si T té valor 0 a posició p, llavors el valor de T' a posici
//       la suma de tots els valors de T a profunditat parell per sobre de p.
BinTree<int> replace0sWithAboveSumDepthEven(BinTree<int> t);
```

Aquí tenim un exemple de comportament de la funció:

```
replace0sWithAboveSumDepthEven(3(0(2,8(0,)),1(6(0,8),8(8,4)))) => 3(3(2,8(11,))
```




Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc`, `BinTree.hh`, `replace0sWithAboveSumDepthEven.hh`. Només cal que creeu `replace0sWithAboveSumDepthEven.cc`, posant-hi els includes que calguin i implementant la funció `replace0sWithAboveSumDepthEven`. Només cal que pugeu `replace0sWithAboveSumDepthEven.cc` al jutge.

### 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

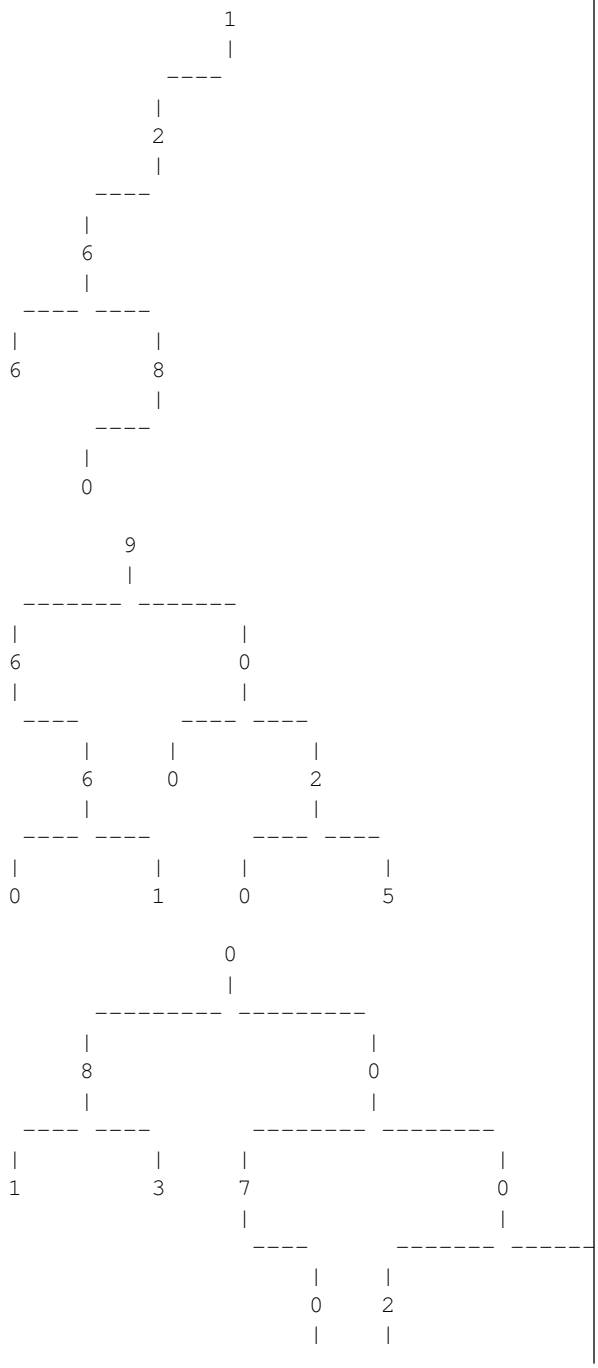
cas consisteix en una descripció d'un arbre un arbre binari d'enters. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

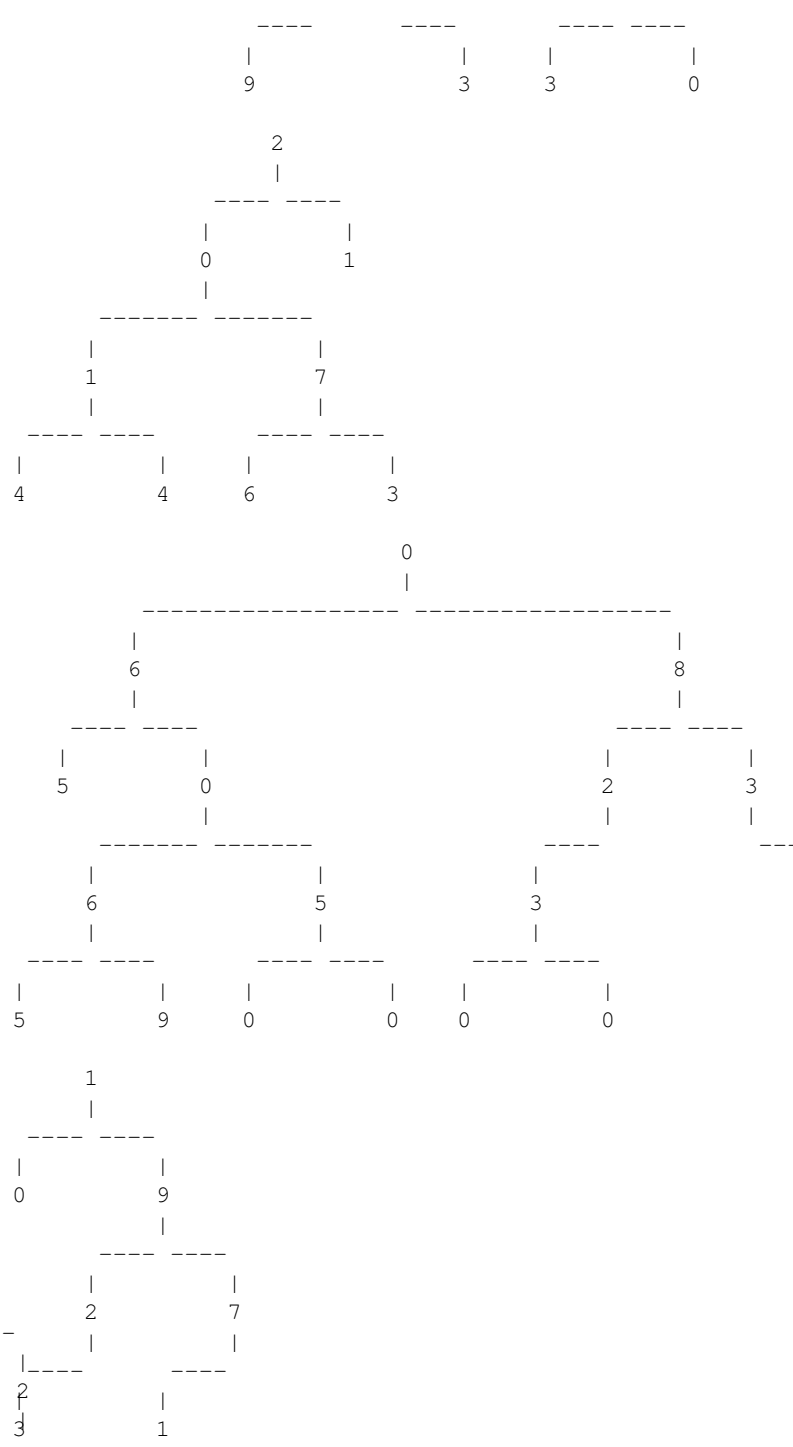
## Sortida

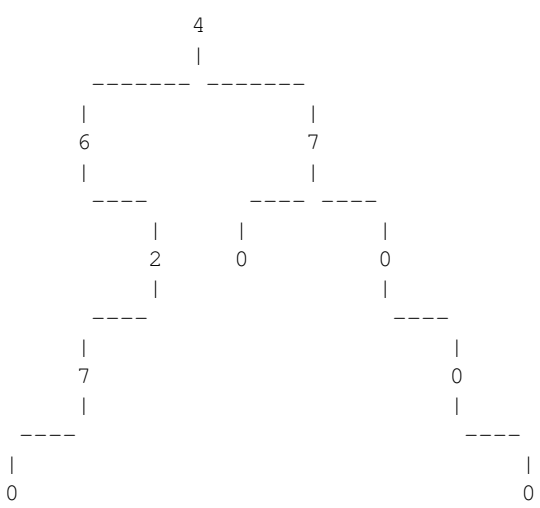
Per a cada cas, cal escriure l'arbre binari resultant de cridar a la funció abans esmentada amb l'arbre d'entrada. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

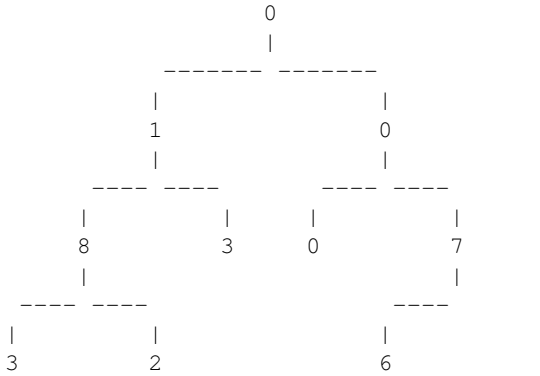
## Exemple d'entrada 1

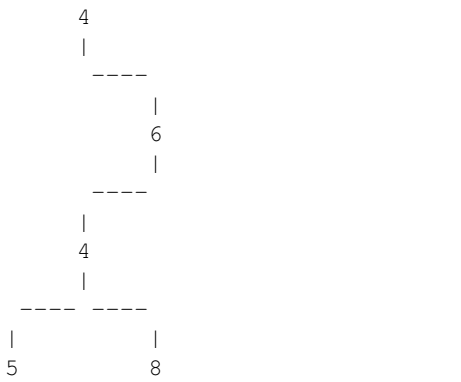
```
VISUALFORMAT
```

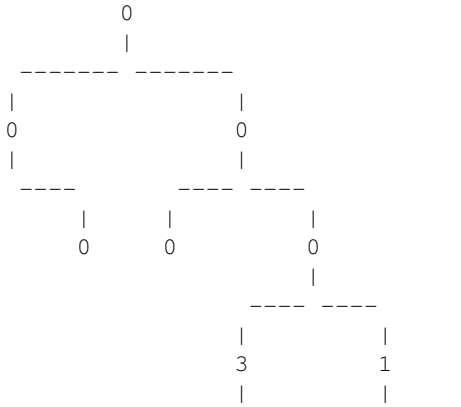


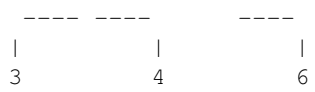




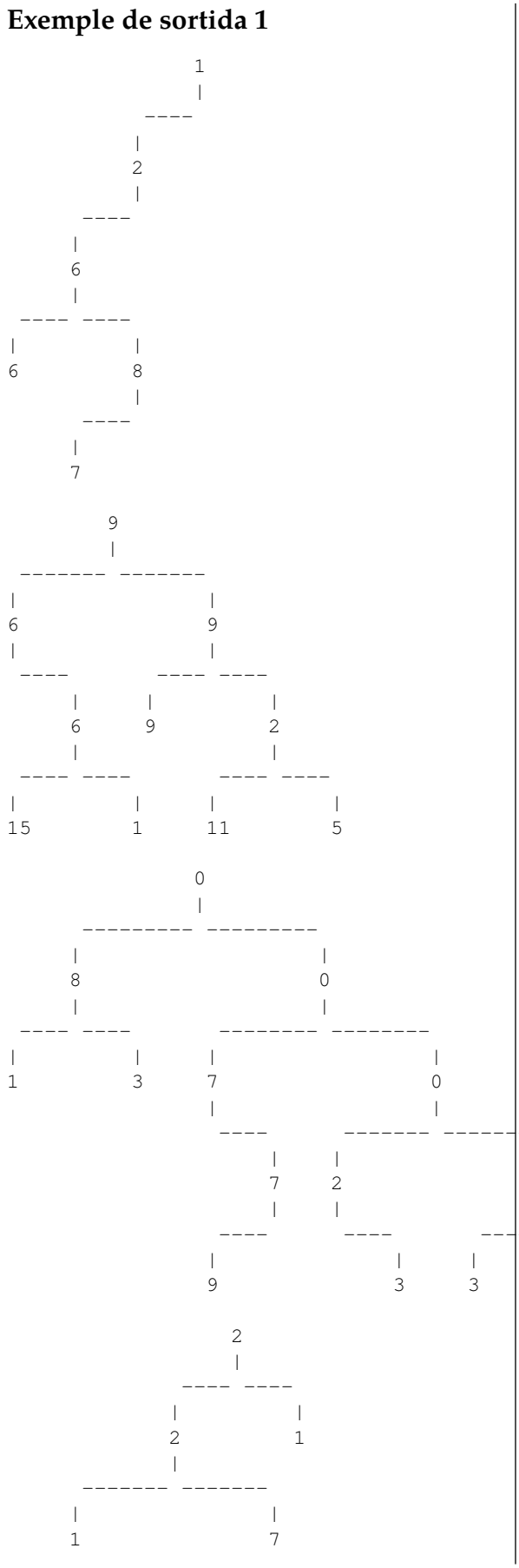


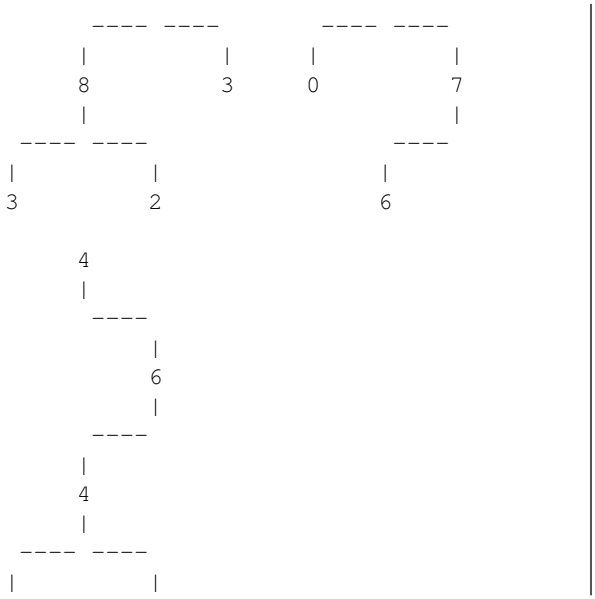


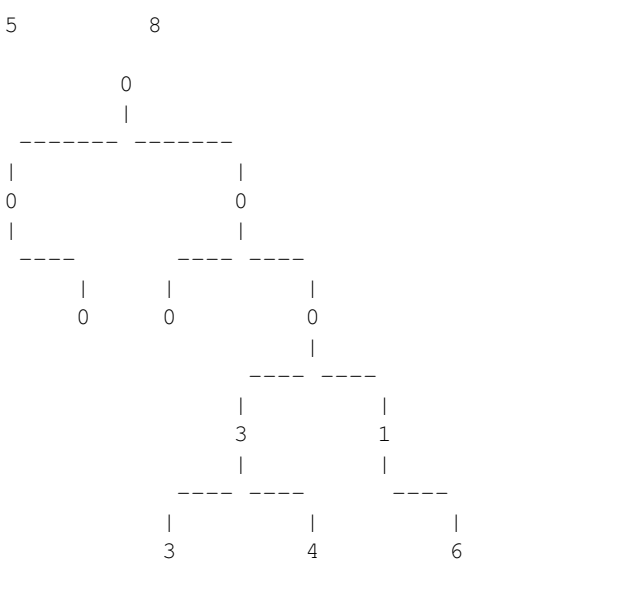




## Exemple de sortida 1







## Exemple d'entrada 2

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

## Exemple de sortida 2

```
1(2(6(6,8(7,)),),)
9(6(,6(15,1)),9(9,2(11,5)))
0(8(1,3),0(7(,7(9,)),0(2(,3),2(3,0))))
2(2(1(4,4),7(6,3)),1)
0(6(5,0(6(5,9),5(0,0))),8(2(3(2,2),),3(,3)))
1(0,9(2(3,),7(1,)))
4(6(,2(7(6,),)),7(4,4(,4(,4))))
0(1(8(3,2),3),0(0,7(6,)))
4(,6(4(5,8),))
0(0(,0),0(0,0(3(3,4),1(,6))))
```

## Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema.

## Informació del problema

Autoria: PRO2

Generació: 2026-01-25T16:48:34.189Z