Reemplaza los nodos de un árbol binario a profundidad par por la suma por debajo — Y97108_es

Implementad una función **RECURSIVA** que, dado un árbol binario de enteros, retorna un nuevo árbol con la misma estructura, y donde cada posición a profundidad par contiene la suma de nodos del subárbol que cuelga de esa misma posición en el árbol inicial, y en cada posición a profundidad impar hay exactamente el mismo valor que se encuentra en esa posición en el árbol inicial.
Sobreentendemos que la raíz del árbol está a profundidad 0, los nodos directos desde la raíz están a profundidad 1, los nodos a distancia dos de la raíz están a profundidad 2, y así sucesivamente. Esta es la cabecera:

```
/**
 * @brief Retorna l'arbre binari `t` reemplaçant els valors dels nodes
 * a profunditat parell per la suma per sota
 *
 * @param t L'arbre binari original.
 *
 * @returns Un arbre binari R amb la mateixa estructura que t.
 * Per a cada posició p de t i R, si p és a profunditat senar,
 * llavors t i R tenen el mateix valor a posició p.
 * En canvi, si p es a profunditat parell, llavors el valor de R a posició
 * p és la suma de tots els valors que es troben a t a posició p i per sota.
 */
BinTree<int> sum_below_at_even_depth(BinTree<int> t);
```

Observación

Los ficheros públicos (icono del gatito) son: la clase `BinTree` (fichero `bintree.hh`), la entrada/salida de `BinTree` (`bintree-io.hh` y `bintree-inline.hh`) y el programa principal. También hay un `Makefile` y el directorio `.vscode` que tiene la configuración para compilar y depurar con VSCode.
Debes implementar `sum_below_at_even_depth` en un **fichero `.cc` nuevo**, compilar (está preparado para poder compilar y depurar con VSCode), y finalmente **enviar solo el fichero con la función**.
Vuestra función y subfunciones que creéis deben trabajar solo con árboles. Muy posiblemente, una solución recursiva directa será lenta, y necesitaréis crear alguna función recursiva auxiliar para producir una solución más eficiente capaz de superar todos los juegos de pruebas.

Entrada

Cada caso consiste en una representación textual de un árbol binario de enteros. (Esta lectura ya la hace el programa principal.)

Salida

Para cada caso, la salida contiene la representación textual del árbol resultante de aplicar la función `sum_below_at_even_depth`. (La salida también la hace el programa principal.)

Ejemplo de entrada 1

```
visual
8
|-- 6
|   |-- 4
|   |   |-- 6
|   |   |   |-- 8
|   |   |   '-- 7
|   |   '-- #
|   '-- 2
|       |-- 7
|       |   |-- 9
|       |   '-- 2
|       '-- 1
|           |-- #
|           '-- 6
'-- 8
    |-- 7
    '-- 3

3
|-- 3
|   |-- 9
|   |   |-- 6
|   |   '-- #
|   '-- 5
'-- 6
    |-- #
    '-- 9
        |-- 5
        '-- #

1
|-- #
'-- 7
    |-- 7
    |   |-- 7
    |   '-- #
    '-- 6
        |-- 3
        |   |-- 7
        |   '-- #
        '-- #

2
|-- #
'-- 1
    |-- 1
    |   |-- #
    |   '-- 2
    '-- #

3
|-- 8
|   |-- 3
|   |   '-- #
'-- 7
    |-- 6
    '-- 1
        |-- 3
        |   |-- 1
        |   '-- 8
        '-- 7

6
|-- 4
|   |-- 4
|   |   |-- 1
|   |   |   |-- 5
|   |   |   '-- #
|   |   '-- 3
|   |       |-- 3
|   |       '-- #
|   '-- #
'-- 3
    |-- 9
    |   |-- 6
    |   |   |-- 9
    |   |   '-- 8
    |   '-- 7
    |       |-- 5
    |       '-- 5
    '-- 2
        |-- 4
        |   |-- 8
        |   '-- 4
        '-- 3
            |-- 7
            '-- 2

3
|-- 8
'-- 1
    |-- 3
    |   |-- #
    |   '-- 4
    |       |-- 9
    |       '-- #
    '-- 3

3
|-- 4
|   |-- 9
|   |   |-- 1
|   |   '-- 6
|   '-- 6
|       |-- #
|       '-- 6
|           |-- #
|           '-- 9
|
```

```
'-- 8
    |-- 8
    '-- 9
        |-- 7
        '-- 2
            |-- #
            '-- 6

1
|-- #
'-- 6
    |-- #
    '-- 3
        |-- 9
        '-- 8

8
|-- 5
|   |-- 2
|   |   |-- #
|   |   '-- 8
|   |       |-- #
|   |       '-- 3
|   '-- 3
|       |-- #
|       '-- 7
|           |-- 7
|           '-- 6
'-- 2
```

Ejemplo de salida 1

```
84
|-- 6
|   |-- 25
|   |   |-- 6
|   |   |   |-- 8
|   |   |   '-- 7
|   |   '-- #
|   '-- 27
|       |-- 7
|       |   |-- 9
|       |   '-- 2
|       '-- 1
|           |-- #
|           '-- 6
'-- 8
    |-- 7
    '-- 3

46
|-- 3
|   |-- 15
|   |   |-- 6
|   |   '-- #
|   '-- 5
'-- 6
    |-- #
    '-- 14
        |-- 5
        '-- #

38
|-- #
'-- 7
    |-- 14
    |   |-- 7
    |   '-- #
    '-- 16
        |-- 3
        |   |-- 7
        |   '-- #
        '-- #

6
|-- #
'-- 1
    |-- 3
    |   |-- #
    |   '-- 2
    '-- #

47
|-- 8
|   |-- 3
|   '-- #
'-- 7
    |-- 6
    '-- 20
        |-- 3
        |   |-- 1
```

```
        |   '-- 8
        '-- 7


108
|-- 4
|   |-- 16
|   |   |-- 1
|   |   |   |-- 5
|   |   |   '-- #
|   |   '-- 3
|   |       |-- 3
|   |       '-- #
|   '-- #
'-- 3
    |-- 49
    |   |-- 6
    |   |   |-- 9
    |   |   '-- 8
    |   '-- 7
    |       |-- 5
    |       '-- 5
    '-- 30
        |-- 4
        |   |-- 8
        |   '-- 4
        '-- 3
            |-- 7
            '-- 2


31
|-- 8
'-- 1
    |-- 16
    |   |-- #
    |   '-- 4
    |       |-- 9
    |       '-- #
    '-- 3


84
```

```
|-- 4
|   |-- 16
|   |   |-- 1
|   |   '-- 6
|   '-- 21
|       |-- #
|       '-- 6
|           |-- #
|           '-- 9
'-- 8
    |-- 8
    '-- 24
        |-- 7
        '-- 2
            |-- #
            '-- 6


27
|-- #
'-- 6
    |-- #
    '-- 20
        |-- 9
        '-- 8


51
|-- 5
|   |-- 13
|   |   |-- #
|   |   '-- 8
|   |       |-- #
|   |       '-- 3
|   '-- 23
|       |-- #
|       '-- 7
|           |-- 7
|           '-- 6
'-- 2
```

Ejemplo de entrada 2

```
inline
8(6(4(6(8,7),),2(7(9,2),1(,6))),8(7,3))
3(3(9(6,),5),6(,9(5,)))
1(,7(7(7,),6(3(7,),)))
2(,1(1(,2),))
3(8(3,),7(6,1(3(1,8),7)))
6(4(4(1(5,),3(3,)),),3(9(6(9,8),7(5,5)),24(8,1(36,19))))
3(8,1(3(,4(9,)),3))
3(4(9(1,6),6(,6(,9))),8(8,9(7,2(,6))))
1(,6(,3(9,8)))
8(5(2(,8(,3)),3(,7(7,6))),2)
```

Ejemplo de salida 2

```
84(6(25(6(8,7),),27(7(9,2),1(,6))),8(7,3))
46(3(15(6,),5),6(,14(5,)))
38(,7(14(7,),16(3(7,),)))
6(,1(3(,2),))
47(8(3,),7(6,20(3(1,8),7)))
108(4(16(1(5,),3(3,)),),3(49(6(9,8),7(5,5)),30(4(8,4),3
84(4(16(1,6),21(,6(,9))),8(8,24(7,2(,6))))
27(,6(,20(9,8)))
51(5(13(,8(,3)),23(,7(7,6))),2)
```

Información del problema

Autoría: PRO2
Traducción: Pau Fernández