

Suma dels valors d'un arbre a profunditat parell

X91811_ca

Implementeu una funció **RECURSIVA** que, donat un arbre binari d'enters, retorna la suma dels seus valors que es troben a profunditat parell. L'arrel es troba a profunditat 0 (parell), els nodes dels seus fills a profunditat 1 (no parell), i així successivament. Aquesta és la capcelera:

```
// Pre:
// Post: Retorna la suma dels valors de t a profunditat parell
int sumAtDepthEven(BinTree<int> t);
```

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

```
sumAtDepthEven (      3      ) => 9

      |
  -----
  |           |
  1           4
  |           |
  -----
      |       |
      5       1
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc`, `BinTree.hh`, `sumAtDepthEven.hh`. Us falta crear el fitxer `sumAtDepthEven.cc` amb els corresponents `includes` i implementar-hi la funció anterior. Només cal que pugueu `sumAtDepthEven.cc` al jutge.

Entrada

La primera línia de l'entrada descriu el format en el que es descriuen els arbres, o bé `IN-LINEFORMAT` 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.

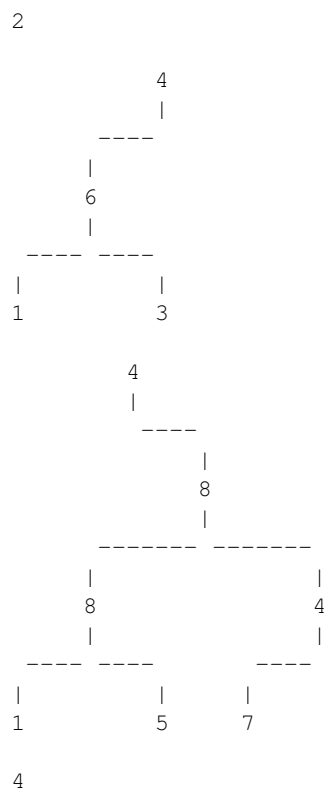
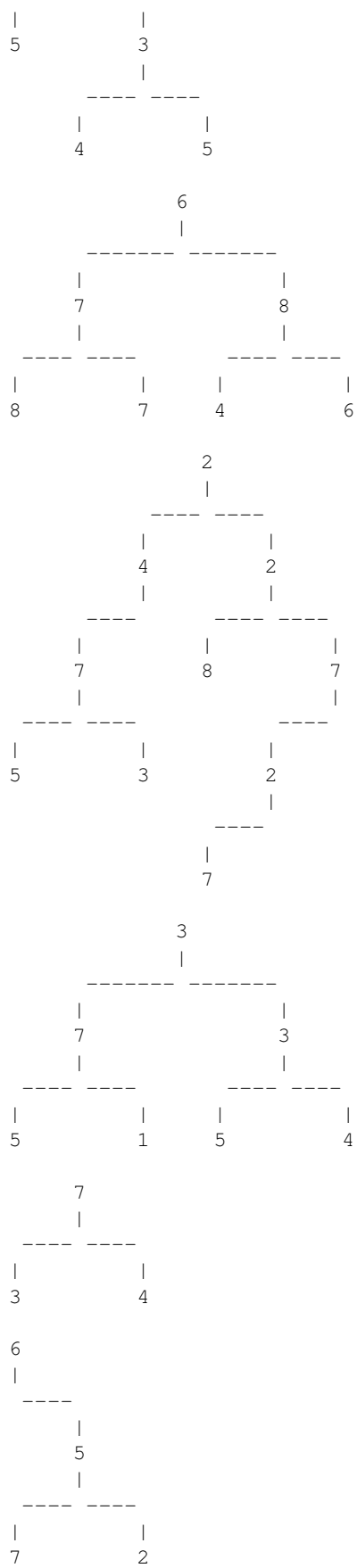
Sortida

Per a cada cas, la sortida conté la corresponent suma dels nodes de l'arbre a profunditat parell. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta suma. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada 1

```
VISUALFORMAT
7
|
```

```
      -----
      |           |
      2           1
      |
      -----
```



Exemple de sortida 1

15
31
31
18

7
15
2
8
16
4

Exemple d'entrada 2

```
INLINEFORMAT
0 (55 (29, -47 (-15, 98) ), -18)
-94 (82 (-21, 80) , -16 (63, -85) )
-27 (-50 (6 (13, -56) , ) , 23 (2, 36 (-2 (-37, ) , ) ) )
-56 (-5 (-100, -37) , 7 (-70, -18) )
5 (-3, -32)
50 (, -23 (-17, 91) )
41
91 (59 (75, -46) , )
55 (, 62 (-31 (-10, 69) , -74 (67, ) ) )
-56
12 (96 (-22 (88, ) , 31 (15, -92) ) , -47 (70, ) )
-58 (4, -1 (27, -35) )
78
-91 (89 (35 (-95, -24) , -50 (, 77) ) , -95)
-69
89 (-93 (, -72) , -31 (-76, -91) )
-25 (93, 76)
32 (-71, 73 (-68 (, -12 (, -70) ) , -86 (-61 (-68, 58) , 150) ) )
68 (-10 (22, 60) , 91)
89 (-7 (-20, 37) , )
```

Exemple de sortida 2

-18
-57
-20
-281
5
124
41
120
-50
-56
91
-66
78
-106
-69
-150
-25
-202
150
106

Observació

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

Necessitareu crear alguna funció recursiva auxiliar amb més paràmetres per tal d'abordar més fàcilment el problema i també per a produir una solució més eficient capaç de superar tots els jocs de proves.

Informació del problema

Autoria: PRO2

Generació: 2026-01-25T17:15:38.578Z

© Jutge.org, 2006–2026.

<https://jutge.org>