

## Màxim per nivells d'un arbre binari

X92145\_ca

Implementa una funció **RECURSIVA** que, donat un arbre binari d'enters retorni un vector de mida igual als nivells que té l'arbre amb el màxim dels valors que hi ha a cada nivell. En concret a cada casella del vector trobarem:

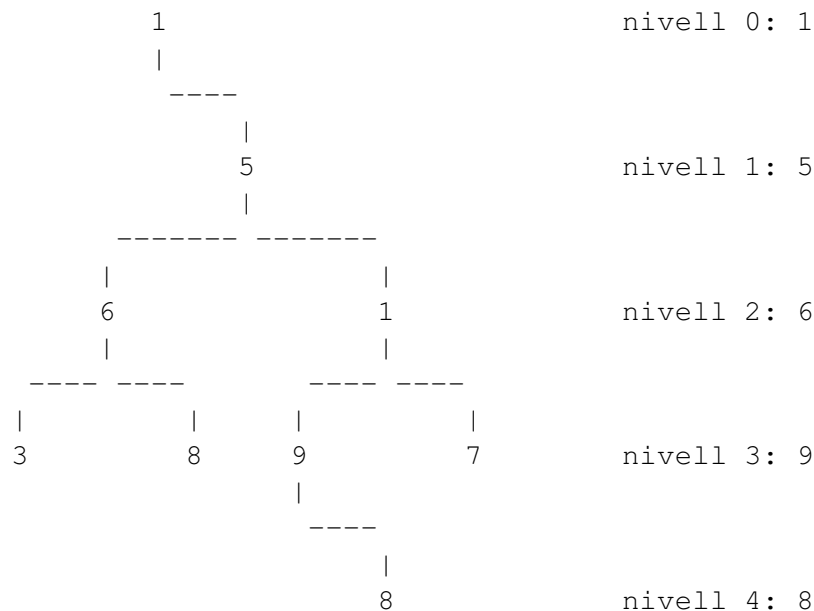
- a la casella 0 hi haurà el màxim dels valors dels nodes amb nivell 0.
- a la casella 1 hi haurà el màxim dels valors dels nodes amb nivell 1.
- etc.

La capçalera de la funció que has d'implementar és la següent:

```
// Pre: cert
// Post: Torna un vector d'enters amb tantes caselles com
// nivells té T i a cada casella hi ha el màxim dels nodes del
// nivell corresponent, és a dir, a la casella i del vector hi
// ha el màxim dels valors dels nodes que estan en el nivell i.
vector<int> maxim_nivells(const BinaryTree<int> &t);
```

Aquí tens un exemple de comportament de la funció:

```
maxim_nivells( 1(, 5(6(3, 8), 1(9(, 8), 7))) ) = [1, 5, 6, 9, 8]
```



Fixa't que l'enunciat d'aquest exercici ja ofereix uns fitxers que has d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `maxim_nivells.hpp`. Només cal que creïs `maxim_nivells.cpp`, posant-hi els includes que calguin i implementant la funció `maxim_nivells`. I quan pugis la teva solució al jutge, només cal que pugis un tar construït així:

```
tar cf solution.tar maxim_nivells.cpp
```

## Entrada

La primera línia 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 binari d'enters. Fixa't que el programa que t'oferim ja s'encarrega de llegir aquestes entrades. **Només cal que implementis la funció abans esmentada.**

## Sortida

Per a cada cas, cal escriure el vector d'enters resultant de cridar a la funció abans esmentada amb l'arbre d'entrada. Fixa't que el programa que t'oferim ja s'encarrega d'escriure aquesta sortida. **Només cal que implementis la funció abans esmentada.**

## Observació

La teva funció i subfuncions que creïs han de treballar només amb arbres binaris i usar la classe `vector` de la biblioteca `STL`. Has de trobar una solució **RECURSIVA** del problema. En les crides recursives, inclou tant la **Hipòtesi d'inducció** com la **funció de fita/decreixement** de cada crida recursiva.

### Exemple d'entrada 1

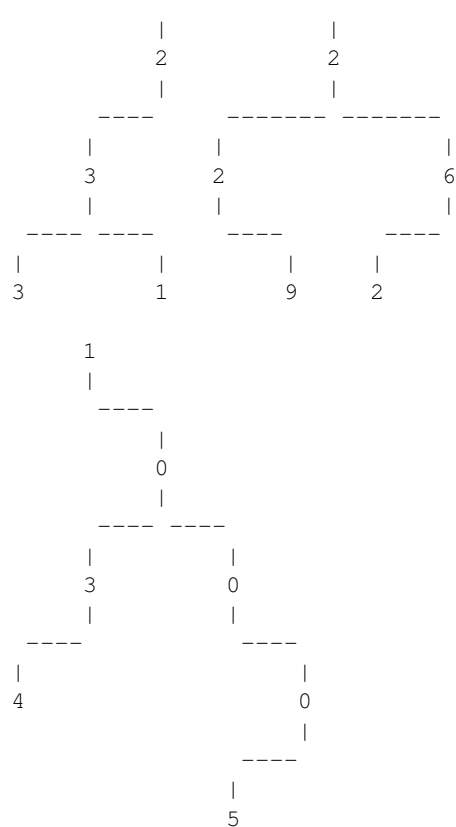
```
INLINEFORMAT
1(2(3,4),5(6(7(8,)),9(10,11))),)
1(,5(6(3,8),1(9(,8),7)))
()
10
```

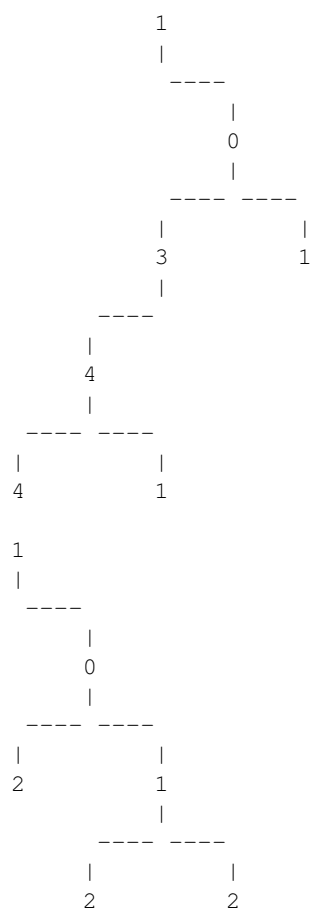
### Exemple de sortida 1

```
[1, 5, 6, 9, 11]
[1, 5, 6, 9, 8]
[]
[10]
```

### Exemple d'entrada 2

```
VISUALFORMAT
      1
      |
      |-----|
          |
          9
          |
          |-----|-----|
          |               |
          2               1
          |               |
          |-----|-----|
          |           |   |   |
          8           2   3   1
                   |
                   |-----|
                           |
                           2
                           |
                           1
                           |
                           |-----|
                               |
                               7
                               |
                               |-----|
```





## Exemple de sortida 2

```

[1, 9, 2, 8, 2]
[1, 7, 2, 6, 9]
[1, 0, 3, 4, 5]
[1, 0, 3, 4, 4]
[1, 0, 2, 2]
  
```

## Informació del problema

Autor : Bernardino Casas  
 Generació : 2024-11-05 11:43:44

© *Jutge.org*, 2006–2024.  
<https://jutge.org>