Antes de pasar a la página siguiente...
¿Has iniciado la grabación del screencast?

Nodos bien colocados en un árbol binario

Segundo parcial - Estructuras de Datos - Grupo F
Facultad de Informática - UCM
Tiene un peso del 20 % en la nota final de la asignatura

Este ejercicio debe entregarse en el siguiente problema *DOMjudge*:

- URL: `http://ed.fdi.ucm.es/`
- Concurso: `ED-F-EV`
- Identificador de problema: `F25`

Debes entregar:

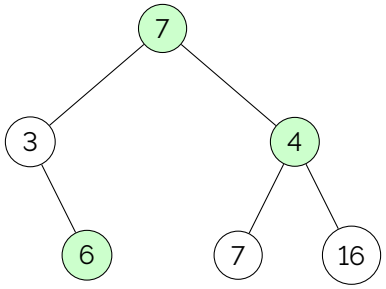
1. El fichero `.cpp` con el código fuente de tu solución. Utiliza la plantilla que se proporciona con este enunciado. El código fuente debe entregarse en *DOMjudge* antes de **1 hora** desde el comienzo del examen. Se evaluará la última entrega realizada antes de la hora límite.
2. Un breve video explicativo (2-5 minutos) de la solución realizada. Este video se entrega a través de la carpeta de *Google Drive* que se ha compartido contigo. Para ello dispones de **30 minutos** una vez finalizado el plazo de entrega del código fuente.
3. El vídeo con el *screencast* de la realización de la prueba. Este video se entrega a través de la carpeta de *Google Drive* que se ha compartido contigo. Para ello dispones de **1 hora** una vez finalizado el plazo de entrega del código fuente.

Recuerda que **no se permite copiar** en la entrega **código fuente externo**, salvo que provenga de la plantilla proporcionada.

Enlace a las instrucciones:
`https://drive.google.com/open?id=1y50P6GtroTeru8-f3TDjn6d8fasIW_9sHNLvMaI5aKs`.

En un árbol binario, el *nivel de profundidad* un nodo es el número de nodos que hay que visitar hasta llegar hasta él. Por ejemplo, la raíz de un árbol está en el nivel 1, los hijos de la raíz están en el nivel 2, los nietos de la raíz están en el nivel 3, y así sucesivamente.

En un árbol binario de números enteros, decimos que un nodo está *bien colocado* si su valor es divisible por el nivel de profundidad en el que está. Por ejemplo, dado el siguiente árbol:



El nodo 7 está bien colocado, ya que es 7 divisible por 1. El nodo 4 está bien colocado, porque 4 es divisible por 2, y el nodo 6 también lo está, pues 6 es divisible por 3. El resto de nodos no cumplen la propiedad de estar bien colocados.

En este ejercicio se pide:

1. Definir una función `nodo_bien_colocado` con la siguiente cabecera:

   ```
   int nodo_bien_colocado(const bintree<int> &tree);
   ```

   Esta función debe devolver el valor del nodo bien colocado dentro de `tree` que esté a mayor nivel de profundidad. En el ejemplo anterior, el nodo bien colocado a mayor profundidad es el 6. En caso de que varios nodos se encuentren en un mismo nivel de profundidad, tendrá prioridad el que se encuentre más a la izquierda. Si el árbol no contiene ningún nodo bien colocado, la función debe devolver `-1`.

2. Indicar el coste, en el caso peor, de la función anterior. El coste debe estar expresado en función del número de nodos del árbol de entrada.

Entrada

La entrada comienza con un número que indica el número de casos de prueba que vienen a continuación. Cada caso de prueba consiste en una línea con la descripción de un árbol binario: primero la raíz (un número entero no negativo), y a continuación la descripción del hijo izquierdo y después la del hijo derecho. El número -1 indica el árbol vacío.

Salida

Para cada árbol se escribirá el valor devuelto por la función `nodo_bien_colocado`.

Entrada de ejemplo

```
3
7 3 -1 6 -1 -1 4 7 -1 -1 16 -1 -1
3 1 -1 -1 7 4 -1 -1 -1
1 2 -1 -1 4 -1 -1
```

Salida de ejemplo

```
6
3
2
```