Junio 2018

2. (2 puntos) En el desarrollo de un juego de aventuras conversacionales se está representando el mapa del juego mediante un árbol binario de personajes. En cada nodo del árbol hay un personaje (*monstruo*, *caballero* o *dama*) representado a través de un carácter (`'M'`,`'C'` y `'D'`, respectivamente)

Se dice que un nodo está *a salvo* cuando: (i) en él hay una dama; (ii) el número de monstruos que hay en el camino que va desde la raiz a dicho nodo es menor o igual que el número de caballeros que hay en los nodos descendientes de dicho nodo.

Debes implementar una función:

```
int num_a_salvo(const Arbin<char>& mapa);
```

que determine el número de nodos en `mapa` que están *a salvo*. Indica además su complejidad.

Septiembre 2018

2. (2 puntos) Un nodo en un árbol binario de enteros se dice que es "curioso" cuando su valor coincide con el resultado de sumar su nivel al número de nodos de su hijo izquierdo. Se debe programar una función

```
int num_curiosos(const Arbin<int>& a)
```

que devuelva el número de nodos "curiosos" que contiene el árbol `a` dado como parámetro. Debe, además, determinarse justificadamente la complejidad de dicha función.

Extraordinario Febrero 2019

2. (2 puntos) En el desarrollo de un juego de aventuras conversacionales se está representando el mapa del juego mediante un árbol binario de personajes. En cada nodo del árbol hay un personaje (*monstruo*, *caballero* o *princesa*) representado a través de un carácter (`'M'`,`'C'` y `'P'`, respectivamente)

Se dice que un nodo está *a salvo* cuando: (i) en él hay una princesa; y (ii) en el camino que va desde la raíz hasta dicho nodo el número de nodos que contienen a un caballero es mayor o igual que el número de nodos que contienen a un monstruo.

Debe implementarse una función:

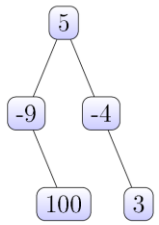
```
int num_a_salvo(const Arbin<char>& mapa);
```

que determine el número de nodos en `mapa` que están *a salvo*. Indica además su complejidad.

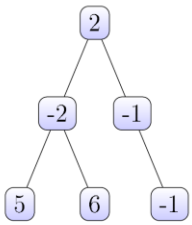
Mayo 2019

2. (2.5 puntos) Un "árbol de ganancias y pérdidas" es un árbol binario de enteros en el que los valores positivos de los nodos representan ganancias, mientras que los negativos representan pérdidas. Un nodo es "rentable" cuando todos sus ancestros son rentables, y, además, la suma de los valores de todos sus ancestros y el del suyo propio es positiva (dicha suma se denomina la "renta" del nodo). El "árbol de ganancias y pérdidas" es "rentable" cuando tiene alguna hoja "rentable" (la "renta" de dicha hoja se denomina una "renta" del árbol).

Ejemplos:



Arbol "rentable"; "renta" máxima: 4



Arbol no "rentable"

Debes programar el procedimiento:

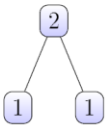
```
void mejor_renta(Arbin<int> a,  bool & es_rentable, int & renta_maxima)
```

que determine: (i) si el "árbol de ganancias y pérdidas" `a` es "rentable" o no; y, en caso positivo, (ii) determine la "renta" máxima del árbol. Si el árbol es "rentable", tras finalizar la ejecución el parámetro `es_rentable` valdrá `true`, y `renta_maxima` contendrá la "renta" máxima del árbol (es decir, la mayor "renta" de las hojas que son rentables). Si el árbol no es "rentable", tras finalizar la ejecución el parámetro `es_rentable` valdrá `false`. En este caso, el valor de `renta_maxima` es irrelevante. También debes determinar justificadamente la complejidad del procedimiento.

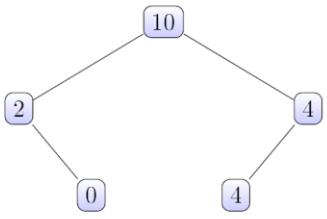
Julio 2019

1. (3.5 puntos) Un nodo interno de un árbol binario de enteros se dice que es *sumativo* si su valor es igual a la suma de los valores de sus descendientes.

A modo de ejemplo, un árbol vacío o un árbol hoja tendrán 0 nodos internos sumativos. El árbol



tendrá un nodo interno sumativo. Y el árbol



tendrá dos nodos internos sumativos.

Implementa la función

```
unsigned int numNodosSumativos (Arbin<int> a)
```

que devuelva el número de nodos internos sumativos que tiene el árbol dado. Determina así mismo de forma justificada la complejidad de dicha función.