Programación Declarativa
2008-2009
Práctica 7

Si se considera que ***Empty*** representa el arbol vacío y que ***Node (r,i,d)*** representa el árbol que tiene raíz ***r***, rama izquierda ***i*** y rama derecha ***d***, el tipo de dato ***'a bintree***, definido a continuación, serviría para representar árboles binarios cuyos nodos están etiquetados con valores de tipo 'a.

```
type 'a bintree = Empty | Node of 'a * 'a bintree * 'a bintree;;
```

Defina en ocaml las siguientes funciones:

`raiz: 'a bintree -> 'a`
que devuelva el valor de la raíz del árbol

`ramaizda: 'a bintree -> 'a bintree`
la rama izquierda del árbol

`ramadcha: 'a bintree -> 'a bintree`
la rama derecha del árbol

`altura: 'a bintree -> int`
el número de níveles del árbol (atención: áltura del árbol vacío = 0)

`num_nodos: 'a bintree -> int`
el número de nodos

`t_max: 'a bintree -> 'a`
el valor máximo de los nodos

`t_min: 'a bintree -> 'a`
el valor mínimo de los nodos

`t_sum: int bintree -> int`
la suma de todos los nodos

`t_spec: 'a bintree ->'a bintree`
la imágen especular del árbol

`hojas: 'a bintree -> 'a list`
la lista de hojas del árbol, de izda a derecha

`monticulo:('a ->'a -> bool) -> 'a bintree -> bool`
si un árbol es "montículo" con una relación de orden dada

`completo: 'a bintree -> bool`
si todas las hojas del árbol están en el mismo nivel

`preorden: 'a bintree -> 'a list`
todos los nodos del árbol en pre-orden

`postorden: 'a bintree -> 'a list`

todos los nodos del árbol en post-orden

`inorden: 'a bintree -> 'a list`
todos los nodos del árbol en in-orden

`anchura: 'a bintree -> 'a list`
todos los nodos del árbol según un recorrido "en achura" de arriba a abajo y de izquierda a derecha.

Considerando el tipo de dato ***'a arbolgen*** definido en la Práctica 6, defina una función **`arbolgen_of_bintree:'a bintree ->'a arbolgen`** que transforme un valor de tipo ***'a bintree*** en el correspondiente de tipo ***'a arbolgen***.

Defina una función **`bintree_of_arbolgen:'a arbolgen ->'a bintree`** que realice la transformación inversa, cuando tenga sentido.