PROGRAMACIÓN DECLARATIVA

10 DE DECEMBRO DE 2010

APELIDOS:_____NOME: _____

E.I. ☐ E.T.I.X. ☐

1. ***(5 puntos)*** Escriba o resultado da compilación e execución das seguintes frases, con tipos e valores, como faría o compilador interactivo de *ocaml*:

```
let five _ = 5;;
```

```
let id x = x and apply x y = x y;;
```

```
five 0, id 0, (id five) 0, id (five 0), apply five true;;
```

```
let mx3 x y z = max x (max y z);;
```

```
mx3 1, mx3 1 2, mx3 1 2 3;;
```

```
let rec fold x = function [] -> x | (op,y)::t -> fold (op x y) t;;
```

```
fold 1 [(+), 2; ( * ), 3; (-), 1; (/), 3];;
```

```
let (|>) x f = f x;;
```

```
-2 |> abs |> (+) 3 |> function x -> x * x;;
```

```
let x = let x = 1 and y = 2 in x + y, x - y;;
```

2. ***(1 punto)*** Reescriba as seguintes definicións sen utilizar definicións locais nin expresións i*f...then...else...*

```
let f x = let (x,y) = x in x;;
```

```
let n x g = if g x then true else false;;
```

3. ***(3 puntos)*** Indique o tipo das seguintes funcións:

```
let rec sorted = function
    [] | [_] -> true
  | x::y::z -> x <= y && sorted (y::z);;
```

```
let rec merge l1 l2 = match (l1,l2) with
    (l, []) | ([], l) -> l
  | (h1::t1, h2::t2) -> if h1 <= h2 then h1 :: merge t1 l2
                        else h2 :: merge l1 t2;;
```

¿É terminal a recursividade empregada nestas definicións?

Se nalgún caso non é así, realice unha nova definición recursiva terminal para a función correspondente.

4. ***(1 punto)*** Considere a seguinte definición en ocaml para o tipo de dato ***bitree*** (que serve para representar árbores binarias en ocaml):

```
type bitree = Empty | Node of bitree * bitree;;
```

Dicimos que unha árbore binaria é "perfecta" se ten cheos todos os seus niveis. Isto é, unha árbore binaria perfecta terá 2^i nodos no nivel *i* (para cada nivel *i* da árbore). Defina a función ***es_perfecta: bitree -> bool*** que indique se unha árbore é ou non é perfecta.