

PROGRAMACIÓN FUNCIONAL

17 de Enero de 2011

1.- (1 punto) ¿Cuál es el tipo más general de las dos siguientes funciones?

- a) `f1 xs ys = head (zipWith (/=) xs ys)`
- b) `f2 g xs = map g (filter (< (head xs)) xs)`

Indicación: No olvides poner las restricciones de clase.

2.- (1 punto) Describir como se evalúa la expresión `xs` (definida abajo) hasta obtener los cinco primeros elementos de la lista resultado.

```
xs = 1 : f xs
    where f (z:zs) = z : f (map (+4) zs)
```

3. (2 puntos) El *código binario reflejado* o *código Gray*, nombrado así en honor del investigador Alessandro Frank Gray, es un sistema de numeración binario en el que dos valores sucesivos difieren solamente en uno de sus dígitos. El código Gray de n -bits es una secuencia de strings de n -bits construidos como sigue:

```
n = 1: ["0", "1"]
n = 2: ["00", "01", "11", "10"]
n = 3: ["000", "001", "011", "010", "110", "111", "101", "100"]
n = 4:
["0000", "0001", "0011", "0010", "0110", "0111", "0101", "0100",
 "1100", "1101", "1111", "1110", "1010", "1011", "1001", "1000"]
etc.
```

Definir una función `gray :: Int -> [String]` tal que

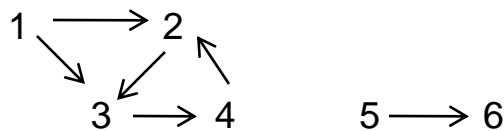
```
> gray 3
["000", "001", "011", "010", "110", "111", "101", "100"]
```

Indicación: Usa inducción sobre n y haz que `gray n` se calcule una sola vez para cada n .

4. (3 puntos) Supongamos que representamos un grafo dirigido como una lista de arcos o pares de nodos. Por ejemplo, la lista de pares

$[(1, 2), (2, 3), (1, 3), (3, 4), (4, 2), (5, 6)]$

representa el grafo:



- a) Definir una función `quitarArco` que dado un grafo g (lista de arcos) y dos nodos x e y , elimine el arco (x, y) del grafo g . Por ejemplo:

```
> quitarArco 3 4 [(1,2), (2,3), (1,3), (3,4), (4,2), (5,6)]
[(1,2), (2,3), (1,3), (4,2), (5,6)]
> quitarArco 4 5 [(1,2), (2,3), (1,3), (3,4), (4,2), (5,6)]
[(1,2), (2,3), (1,3), (3,4), (4,2), (5,6)]
```

- b) ¿Cuál es el tipo más general de `quitarArco`?

- c) Definir una función que dado un grafo g (lista de arcos) y dos nodos x e y , decida si existe un camino que va de x a y en g . Por ejemplo:

```
> hayCamino [(1,2), (2,3), (1,3), (3,4), (4,2), (5,6)] 1 4
True
> hayCamino [(1,2), (2,3), (1,3), (3,4), (4,2), (5,6)] 2 6
False
```

Indicación: Utiliza la función `quitarArco` para evitar entrar en un ciclo cuando no existe el camino de x a y

- d) ¿Cuál es el tipo más general de la función `hayCamino`?