Informática de 1º de Matemáticas (Grupo 1) **(30–Noviembre–2009)**

Ejercicio 1 (2 puntos) Definir por recursión la función

```
sumaFactR :: Int -> Int
```

tal que `(sumaFactR n)` es la suma de los factoriales de los números desde 0 hasta `n`. Por ejemplo,

```
sumaFactR 3  =>  10
```

Solución:

```
sumaFactR :: Int -> Int
sumaFactR 0     = 1
sumaFactR (n+1) = factorial (n+1) + sumaFactR n

factorial n = product [1..n]
```

Ejercicio 2 (2 puntos) Definir por comprensión la función

```
sumaFactC :: Int -> Int
```

tal que `(sumaFactC n)` es la suma de los factoriales de los números desde 0 hasta `n`. Por ejemplo,

```
sumaFactC 3  =>  10
```

Solución:

```
sumaFactC :: Int -> Int
sumaFactC n = sum [factorial x | x <- [0..n]]
```

Ejercicio 3 (2 puntos) Definir la función

```
copia :: [a] -> Int -> [a]
```

tal que `(copia xs n)` es la lista obtenida copiando `n` veces la lista `xs`. Por ejemplo,

```
copia "abc" 3  =>  "abcabcabc"
```

Solución:

```
copia :: [a] -> Int -> [a]
copia xs 0     = []
copia xs (n+1) = xs ++ copia xs n
```

Ejercicio 4 (2 puntos) Definir por recursión la función

```
incidenciasR :: Eq a => a -> [a] -> Int
```

tal que `(incidenciasR x ys)` es el número de veces que aparece el elemento `x` en la lista `ys`. Por ejemplo,

```
incidenciasR 3 [7,3,5,3]  =>  2
```

Solución:

```
incidenciasR :: Eq a => a -> [a] -> Int
incidenciasR _ []                 = 0
incidenciasR x (y:ys) | x == y    = 1 + incidenciasR x ys
                      | otherwise = incidenciasR x ys
```

Ejercicio 5 (2 puntos) Definir por comprensión la función

```
incidenciasC :: Eq a => a -> [a] -> Int
```

tal que `(incidenciasC x ys)` es el número de veces que aparece el elemento `x` en la lista `ys`. Por ejemplo,

```
incidenciasC 3 [7,3,5,3]  =>  2
```

Solución:

```
incidenciasC :: Eq a => a -> [a] -> Int
incidenciasC x ys = length [y | y <- ys, y == x]
```