EJERCICIOS BLOG

1) Escribir una versión de factorial que funcione también para argumentos negativos; en ese caso deberá ignorar el signo (por ej. el factorial de 6 y el de -6 deben devolver lo mismo).

```
fact :: Int -> Int
fact 0 = 1
fact n = n * fact(n-1)

fact1 :: Int -> Int
fact1 n = fact (abs(n))

fact2 :: Int -> Int
fact2 0 = 1
fact2 n = abs(n) * fact2(abs(n)-1)

fact3 :: Int -> Int
fact3 0 = 1
fact3 n = if n > 0 then n * fact3(n-1) else -n * fact3(n+1)
```

2) Dada una lista de enteros, devolver los enteros que sean pares.

```
pares :: [Int] -> [Int]
pares [] = []
pares (x:y) = if par(x) then x : (pares y) else pares y

par :: Int -> Bool
par 0 = True
par 1 = False
par x = par (x - 2)
```

3) Dadas dos listas (de enteros), devolver el intercalado de ambas.
Ejemplo: la función aplicada a [1,2,3] y [4,5,6,7,8] devolverá [1,4,2,5,3,6,7,8].

```
intercalado :: ([a], [a]) -> [a]
intercalado (z, []) = z
intercalado ([], z) = z
intercalado (x:xs, y:ys) = x : (y : (intercalado(xs,ys)))

intercaladoV2 :: ([a], [a]) -> [a]
intercaladoV2 ([], z) = z
intercaladoV2 (x:y, z) = x : (intercaladoV2(z,y))

-- Intercalado de 3 listas --

intercalado3 :: ([a], [a], [a]) -> [a]
intercalado3 (z, [], []) = z
intercalado3 ([], z, []) = z
intercalado3 ([], [], z) = z
intercalado3 (x, y, []) = intercalado (x,y)
intercalado3 (x, [], z) = intercalado (x,z)
intercalado3 ([], y, z) = intercalado (y,z)
intercalado3 (x:xs, y:ys, z:zs) = x : (y : (z : (intercalado3(xs, ys, zs))))
```