

EJERCICIOS HASKELL

Ejercicio 3.3.1. *El doble factorial de un número n se define por*

$$\begin{aligned}0!! &= 1 \\1!! &= 1 \\n!! &= n*(n-2)* \dots * 3 * 1, \text{ si } n \text{ es impar} \\n!! &= n*(n-2)* \dots * 4 * 2, \text{ si } n \text{ es par}\end{aligned}$$

Por ejemplo,

$$\begin{aligned}8!! &= 8*6*4*2 = 384 \\9!! &= 9*7*5*3*1 = 945\end{aligned}$$

Definir, por recursión, la función

```
dobleFactorial :: Integer -> Integer
```

tal que (dobleFactorial n) es el doble factorial de n . Por ejemplo,

```
dobleFactorial 8 == 384
dobleFactorial 9 == 945
```

Ejercicio 3.1.1. *Definir por recursión la función*

```
potencia :: Integer -> Integer -> Integer
```

tal que (potencia x n) es x elevado al número natural n . Por ejemplo,

```
potencia 2 3 == 8
```

Ejercicio 3.5.1. *Definir por recursión la función*

```
menorDivisible :: Integer -> Integer -> Integer
```

tal que (menorDivisible a b) es el menor número divisible por los números del a al b . Por ejemplo,

```
menorDivisible 2 5 == 60
```

Indicación: Usar la función lcm tal que (lcm x y) es el mínimo común múltiplo de x e y .

Ejercicio 3.14.7. Definir por recursión la función

```
esPermutacion :: Eq a => [a] -> [a] -> Bool
```

tal que (esPermutacion xs ys) se verifica si xs es una permutación de ys. Por ejemplo,

```
esPermutacion [1,2,1] [2,1,1] == True
esPermutacion [1,2,1] [1,2,2] == False
```

Ejercicio 4.1.3. Definir, por comprensión, la función

```
sumaCuadradosC :: Integer -> Integer
```

tal que (sumaCuadradosC n) es la suma de los cuadrados de los números de 1 a n. Por ejemplo,

```
sumaCuadradosC 4 == 30
```

Ejercicio 4.2.1. Se quiere formar una escalera con bloques cuadrados, de forma que tenga un número determinado de escalones. Por ejemplo, una escalera con tres escalones tendría la siguiente forma:

```
  XX
XXXX
XXXXXX
```

Definir, por recursión, la función

```
numeroBloquesR :: Integer -> Integer
```

tal que (numeroBloquesR n) es el número de bloques necesarios para construir una escalera con n escalones. Por ejemplo,

```
numeroBloquesR 1 == 2
numeroBloquesR 3 == 12
numeroBloquesR 10 == 110
```

Ejercicio 4.2.2. Definir, por comprensión, la función

```
numeroBloquesC :: Integer -> Integer
```

tal que (numeroBloquesC n) es el número de bloques necesarios para construir una escalera con n escalones. Por ejemplo,

```
numeroBloquesC 1 == 2
numeroBloquesC 3 == 12
numeroBloquesC 10 == 110
```

Ejercicio 4.4.9. Definir, por recursión, la función

```
listaNumeroR :: [Integer] -> Integer
```

tal que (listaNumeroR xs) es el número formado por los dígitos de la lista xs. Por ejemplo,

```
listaNumeroR [5]          == 5
listaNumeroR [1,3,4,7]    == 1347
listaNumeroR [0,0,1]      == 1
```

Ejercicio 4.4.10. Definir, por comprensión, la función

```
listaNumeroC :: [Integer] -> Integer
```

tal que (listaNumeroC xs) es el número formado por los dígitos de la lista xs. Por ejemplo,

```
listaNumeroC [5]          == 5
listaNumeroC [1,3,4,7]    == 1347
listaNumeroC [0,0,1]      == 1
```