



Ejercicio 1.

a) Implementar la función parcial $f :: \text{Integer} \rightarrow \text{Integer}$ definida por extensión de la siguiente manera:

$$\begin{aligned}f(1) &= 8 \\f(4) &= 131 \\f(16) &= 16\end{aligned}$$

y cuya especificación es:

```
problema f (n : Z) : Z {  
  requiere: {n = 1 ∨ n = 4 ∨ n = 16}  
  asegura: {(n = 1 → res = 8) ∧ (n = 4 → res = 131) ∧ (n = 16 → res = 16)}  
}
```

b) Análogamente, especificar e implementar la función parcial $g :: \text{Integer} \rightarrow \text{Integer}$

$$\begin{aligned}g(8) &= 16 \\g(16) &= 4 \\g(131) &= 1\end{aligned}$$

c) A partir de las funciones definidas en los ítems a) y b), implementar las funciones parciales $h = f \circ g$ y $k = g \circ f$

Ejercicio 2. ★ Especificar e implementar las siguientes funciones, incluyendo su signatura.

- a) **absoluto**: calcula el valor absoluto de un número entero.
- b) **maximoAbsoluto**: devuelve el máximo entre el valor absoluto de dos números enteros.
- c) **maximo3**: devuelve el máximo entre tres números enteros.
- d) **algunoEsCero**: dados dos números racionales, decide si alguno es igual a 0 (resolverlo con y sin *pattern matching*).
- e) **ambosSonCero**: dados dos números racionales, decide si ambos son iguales a 0 (resolverlo con y sin *pattern matching*).
- f) **enMismoIntervalo**: dados dos números reales, indica si están relacionados por la relación de equivalencia en \mathbb{R} cuyas clases de equivalencia son: $(-\infty, 3]$, $(3, 7]$ y $(7, \infty)$, o dicho de otra manera, si pertenecen al mismo intervalo.
- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).
- h) **esMultiploDe**: dados dos números naturales, decide si el primero es múltiplo del segundo.
- i) **digitoUnidades**: dado un número entero, extrae su dígito de las unidades.
- j) **digitoDecenas**: dado un número entero mayor a 9, extrae su dígito de las decenas.

Ejercicio 3. Implementar una función $\text{estanRelacionados} :: \text{Integer} \rightarrow \text{Integer} \rightarrow \text{Bool}$

```
problema estanRelacionados (a : Z, b : Z) : Bool {  
  requiere: {a ≠ 0 ∧ b ≠ 0}  
  asegura: {(res = true) ↔ (a * a + a * b * k = 0 para algún k ∈ Z con k ≠ 0)}  
}
```

Por ejemplo:

$\text{estanRelacionados } 8 \ 2 \rightsquigarrow \text{True}$ porque existe $k = -4$ tal que $8^2 + 8 \times 2 \times (-4) = 0$
 $\text{estanRelacionados } 7 \ 3 \rightsquigarrow \text{False}$ porque no existe un k entero tal que $7^2 + 7 \times 3 \times k = 0$

Ejercicio 4. ★ Especificar e implementar las siguientes funciones utilizando tuplas para representar pares y ternas de números.

- a) **productoInterno**: calcula el producto interno entre dos tuplas de $\mathbb{R} \times \mathbb{R}$.
- b) **esParMenor**: dadas dos tuplas de $\mathbb{R} \times \mathbb{R}$, decide si cada coordenada de la primera tupla es menor a la coordenada correspondiente de la segunda tupla.
- c) **distancia**: calcula la distancia euclídea entre dos puntos de \mathbb{R}^2 .
- d) **sumaTerna**: dada una terna de enteros, calcula la suma de sus tres elementos.
- e) **sumarSoloMultiplos**: dada una terna de números enteros y un natural, calcula la suma de los elementos de la terna que son múltiplos del número natural.
Por ejemplo:
`sumarSoloMultiplos (10,-8,-5) 2 ~> 2`
`sumarSoloMultiplos (66,21,4) 5 ~> 0`
`sumarSoloMultiplos (-30,2,12) 3 ~> -18`
- f) **posPrimerPar**: dada una terna de enteros, devuelve la posición del primer número par si es que hay alguno, o devuelve 4 si son todos impares.
- g) **crearPar** :: `a -> b -> (a, b)`: a partir de dos componentes, crea un par con esos valores. Debe funcionar para elementos de cualquier tipo.
- h) **invertir** :: `(a, b) -> (b, a)`: invierte los elementos del par pasado como parámetro. Debe funcionar para elementos de cualquier tipo.
- i) Reescribir los ejercicios **productoInterno**, **esParMenor** y **distancia** usando el siguiente renombre de tipos:
`type Punto2D = (Float, Float)`

Ejercicio 5. Implementar la función `todosMenores :: (Integer, Integer, Integer) -> Bool`

```
problema todosMenores (t :  $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ ) : Bool {  
    requiere: {True}  
    asegura: {(res = true)  $\leftrightarrow$  ((f(t0) > g(t0))  $\wedge$  (f(t1) > g(t1))  $\wedge$  (f(t2) > g(t2)))}  
}  
  
problema f (n :  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
    requiere: {True}  
    asegura: {(n  $\leq$  7  $\rightarrow$  res = n2)  $\wedge$  (n > 7  $\rightarrow$  res = 2n - 1)}  
}  
  
problema g (n :  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
    requiere: {True}  
    asegura: {Si n es un número par entonces res = n/2, en caso contrario, res = 3n + 1}  
}
```

Ejercicio 6. Usando los siguientes tipos:

```
type Anio = Integer  
type EsBisiesto = Bool
```

Programar la función `bisiesto :: Anio -> EsBisiesto` según la siguiente especificación:

```
problema bisiesto (año :  $\mathbb{Z}$ ) : Bool {  
    requiere: {True}  
    asegura: {(res = false)  $\leftrightarrow$  (año no es múltiplo de 4, o bien, año es múltiplo de 100 pero no de 400)}  
}
```

Por ejemplo:

```
bisiesto 1901 ~> False    bisiesto 1904 ~> True  
bisiesto 1900 ~> False    bisiesto 2000 ~> True
```

Ejercicio 7.

a) Implementar la función:

```
distanciaManhattan :: (Float, Float, Float) -> (Float, Float, Float) -> Float

problema distanciaManhattan (p : ℝ × ℝ × ℝ, q : ℝ × ℝ × ℝ) : ℝ {
  requiere: {True}
  asegura: {res = ∑i=02 |pi - qi|}
}
```

Por ejemplo:

```
distanciaManhattan (2, 3, 4) (7, 3, 8) ~ 9
distanciaManhattan ((-1), 0, (-8.5)) (3.3, 4, (-4)) ~ 12.8
```

b) Reimplementar la función teniendo en cuenta el siguiente tipo: `type Punto3D = (Float, Float, Float)`

Ejercicio 8. Implementar la función `comparar :: Integer -> Integer -> Integer`

```
problema comparar (a : ℤ, b : ℤ) : ℤ {
  requiere: {True}
  asegura: {(res = 1) ↔ (sumaUltimosDosDigitos(a) < sumaUltimosDosDigitos(b))}
  asegura: {(res = -1) ↔ (sumaUltimosDosDigitos(a) > sumaUltimosDosDigitos(b))}
  asegura: {(res = 0) ↔ (sumaUltimosDosDigitos(a) = sumaUltimosDosDigitos(b))}
}
```

```
problema sumaUltimosDosDigitos (x : ℤ) : ℤ {
  requiere: {True}
  asegura: {res = (|x| mód 10) + (⌊|x|/10⌋ mód 10)}
}
```

Por ejemplo:

```
comparar 45 312 ~ -1 porque 45 < 312 y 4 + 5 > 1 + 2.
comparar 2312 7 ~ 1 porque 2312 < 7 y 1 + 2 < 0 + 7.
comparar 45 172 ~ 0 porque no vale 45 < 172 ni tampoco 172 < 45.
```

Ejercicio 9. A partir de las siguientes implementaciones en Haskell, describir en lenguaje natural qué hacen y especificarlas.

- | | |
|--|---|
| a) <code>f1 :: Float -> Float</code>
<code>f1 n n == 0 = 1</code>
<code> otherwise = 0</code> | d) <code>f4 :: Float -> Float -> Float</code>
<code>f4 x y = (x+y)/2</code> |
| b) <code>f2 :: Float -> Float</code>
<code>f2 n n == 1 = 15</code>
<code> n == -1 = -15</code> | e) <code>f5 :: (Float, Float) -> Float</code>
<code>f5 (x, y) = (x+y)/2</code> |
| c) <code>f3 :: Float -> Float</code>
<code>f3 n n <= 9 = 7</code>
<code> n >= 3 = 5</code> | f) <code>f6 :: Float -> Int -> Bool</code>
<code>f6 a b = truncate a == b</code> |