Introducción a la Programación

Algoritmos y Estructuras de Datos I

Segundo cuatrimestre de 2025

Departamento de Computación - FCEyN - UBA

Práctica 3: Introducción a Haskell
Segunda Parte

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

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.

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

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.

Una posible especificación:

`problema` `esParMenor` (t1, t2: $\mathbb{R} \times \mathbb{R}$) : Bool $\{$
  `requiere:` $\{$True$\}$
  `asegura:` $\{$ $result = true \leftrightarrow$ la primera componente de $t1$ es menor que la primera componente de $t2$, y la segunda componente de $t1$ es menor que la segunda componente de $t2\}$
$\}$

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

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.

Una posible especificación:

`problema` `esParMenor` (t1, t2: $\mathbb{R} \times \mathbb{R}$) : Bool $\{$
  `requiere:` $\{$True$\}$
  `asegura:` $\{$ $result = true \leftrightarrow$ la primera componente de $t1$ es menor que la primera componente de $t2$, y la segunda componente de $t1$ es menor que la segunda componente de $t2\}$
$\}$

Modificar la implementación para usar el siguiente tipo:

```
type Punto2D = (Float, Float)
```

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

f) **posPrimerPar:** dada una terna de enteros, devuelve la posición del primer número par si es que hay alguno, y devuelve 4 si son todos impares.

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

f) **posPrimerPar:** dada una terna de enteros, devuelve la posición del primer número par si es que hay alguno, y devuelve 4 si son todos impares.

Una posible especificación:

problema posPrimerPar (t: $\mathbb{Z}\times\mathbb{Z}\times\mathbb{Z}$) : $\mathbb{Z}$ {
  requiere: { True }
  asegura: {si algún componente de t es par, entonces res es un valor entre 1 y 3 (inclusive), y es la posición del primer elemento par}
  asegura: {si ningún componente de t es par, entonces $res = 4$}
}

Ejercicio 4: posPrimerPar

`problema` posPrimerPar (t: $\mathbb{Z}\times\mathbb{Z}\times\mathbb{Z}$) : $\mathbb{Z}$ {
`requiere:` { True }
`asegura:` {si algún componente de t es par, entonces res es un valor entre 1 y 3 (inclusive), y es la posición del primer elemento par}
`asegura:` {si ningún componente de t es par, entonces $res = 4$}
}

```
posPrimerPar :: (Int, Int, Int) -> Int
posPrimerPar (x,y,z) | mod x 2 == 0 = 1
                     | mod y 2 == 0 = 2
                     | mod z 2 == 0 = 3
                     | otherwise = 4
```

```
posPrimerPar :: (Int, Int, Int) -> Int
posPrimerPar (x,y,z) | mod x 2 == 0 = 0
                     | mod y 2 == 0 = 1
                     | mod z 2 == 0 = 2
                     | otherwise = 4
```

Ejercicio 6

Programar una función `bisiesto :: Integer -> Bool` según la siguiente especificación:

```
problema bisiesto (año: ℤ) : Bool {
  requiere: {True}
  asegura: {res=false ↔ año no es múltiplo de 4 o año es múltiplo de
        100 pero no de 400}
}
```

Ejercicio 6

Programar una función `bisiesto :: Integer -> Bool` según la siguiente especificación:

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

Una posible solución:

```
bisiesto :: Int -> Bool
bisiesto x | mod x 100 == 0 = mod x 400 == 0
           | otherwise = mod x 4 == 0
```

Ejercicio 7

Implementar la función:

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

problema distanciaManhattan (p: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$, q: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$) : $\mathbb{R}$ {
  requiere: {True}
  asegura: {res $= \sum_{i=0}^{2} |p_i - q_i|$}
}

Ejercicio 7

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=0}^{2} |p_i − q_i|}
}
```

Reimplementarla teniendo en cuenta el siguiente tipo: `type Coordenada3d = (Float, Float, Float)`

Una posible solución ejercicio 7

problema distanciaManhattan (p: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$, q: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$) : $\mathbb{R}$ {
  requiere: {True}
  asegura: {res $= \sum_{i=0}^{2} |p_i - q_i|$}
}

```
-- Ejercicio 2a Cambio Int por Float
absoluto :: Float -> Float
absoluto n | n < 0 = -n
           |otherwise = n
```

Una posible solución ejercicio 7

problema distanciaManhattan (p: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$, q: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$) : $\mathbb{R}$ {
  requiere: {True}
  asegura: {res $= \sum_{i=0}^{2} |p_i - q_i|$}
}

```
-- Ejercicio 2a Cambio Int por Float
absoluto :: Float -> Float
absoluto n | n < 0 = -n
           |otherwise = n

distanciaManhattan :: (Float, Float, Float) ->
  (Float, Float, Float) -> Float
distanciaManhattan (x1, y1, z1) (x2, y2, z2) =
    absoluto (x1 - x2) + absoluto (y1 - y2)
    + absoluto (z1 - z2)
```

Una posible solución ejercicio 7

problema distanciaManhattan (p: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$, q: $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$) : $\mathbb{R}$ {
  requiere: {True}
  asegura: {res $= \sum_{i=0}^{2} |p_i - q_i|$}
}

```
-- Ejercicio 2a Cambio Int por Float
absoluto :: Float -> Float
absoluto n | n < 0 = -n
           |otherwise = n

distanciaManhattan :: (Float, Float, Float) ->
  (Float, Float, Float) -> Float
distanciaManhattan (x1, y1, z1) (x2, y2, z2) =
    absoluto (x1 - x2) + absoluto (y1 - y2)
    + absoluto (z1 - z2)
```

Reimplementarla teniendo en cuenta el siguiente tipo: `type Coordenada3d = (Float, Float, Float)`

Ejercicio 8

Implementar una 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))}
}
```

Posible solución ejercicio 8

```
absoluto :: Int -> Int --Ejercicio 2a

digitoUnidades :: Int -> Int --Ejercicio 2i

digitoDecenas :: Int -> Int --Ejercicio 2j

comparar :: Int -> Int -> Int
comparar x y | digitoUnidades x + digitoDecenas x <
              digitoUnidades y + digitoDecenas y = 1
             | digitoUnidades x + digitoDecenas x >
              digitoUnidades y + digitoDecenas y = -1
             | otherwise = 0
```

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

d)
```
f4 :: Float -> Float -> Float
f4 x y = (x+y)/2
```

e)
```
f5 :: (Float, Float) -> Float
f5 (x, y) = (x+y)/2
```

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

d)
```
f4 :: Float -> Float -> Float
f4 x y = (x+y)/2
```

e)
```
f5 :: (Float, Float) -> Float
f5 (x, y) = (x+y)/2
```

- ¿Qué hacen estas dos funciones?
- ¿Hacen lo mismo?
- ¿Son **iguales**?

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

d)
```
f4 :: Float -> Float -> Float
f4 x y = (x+y)/2
```

e)
```
f5 :: (Float, Float) -> Float
f5 (x, y) = (x+y)/2
```

- ¿Qué hacen estas dos funciones?
- ¿Hacen lo mismo?
- ¿Son **iguales**?

problema f4 (x,y: $\mathbb{R}$) : $\mathbb{R}$ {
  requiere: {True}
  asegura: {res = (x+y)/2}
}

problema f5 (t: $\mathbb{R} \times \mathbb{R}$) : $\mathbb{R}$ {
  requiere: {True}
  asegura: {res = $(t_0+t_1)/2$}
}