Introducción a la programación

Práctica 3: Introducción a Haskell
Segunda Parte

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

b) **todoMenor:** dadas dos tuplas $\mathbb{R} \times \mathbb{R}$, decide si es cierto que 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) **todoMenor:** dadas dos tuplas $\mathbb{R} \times \mathbb{R}$, decide si es cierto que cada coordenada de la primera tupla es menor a la coordenada correspondiente de la segunda tupla.

problema todoMenor (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

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: ℤ×ℤ×ℤ) : ℤ  {
  requiere: { - }
  asegura: {si algun elemento es par, entonces res es la posición
        del primer elemento par}
  asegura: {si ningún elemento es par, entonces res = 4}
}
```

Ejercicio 4: posPrimerPar

problema posPrimerPar (t: $\mathbb{Z}\times\mathbb{Z}\times\mathbb{Z}$) : $\mathbb{Z}$ {
  requiere: { - }
  asegura: {si algun elemento es par, entonces res es la posición
        del primer elemento par}
  asegura: {si ningún elemento 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 4: posPrimerPar

Una especificación más precisa

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

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

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 semiformalmente.

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 semiformalmente.

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: $\{res = (t_0 + t_1)/2\}$
}