

Introducción a la programación

Práctica 2: Especificación de problemas

Práctica 3: Introducción a Haskell

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- d) problema `raicesCuadradasUno` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
- requiere: {Todos los elementos de s son positivos}
 - requiere: {No hay elementos repetidos en s }
 - asegura: {*resultado* tiene la misma cantidad de elementos que s }
 - asegura: {Los elementos de *resultado* son la salida de aplicar el problema `raizCuadrada()` a todos los elementos de la secuencia s }
 - asegura: {El orden de la secuencia *resultado* es el mismo que en la secuencia s }
- }

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- d) problema `raicesCuadradasUno` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
 asegura: {El orden de la secuencia *resultado* es el mismo
 que en la secuencia s }
}

► $s = \langle 4, 1, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- d) problema `raicesCuadradasUno` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
 asegura: {El orden de la secuencia *resultado* es el mismo
 que en la secuencia s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1, 3 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- d) problema `raicesCuadradasUno` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
 asegura: {El orden de la secuencia *resultado* es el mismo
 que en la secuencia s }
}

► $s = \langle 4, 1, 9 \rangle$, *resultado* = $\langle 2, 1, 3 \rangle$

► $s = \langle 25, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- d) problema `raicesCuadradasUno` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
 asegura: {El orden de la secuencia *resultado* es el mismo
 que en la secuencia s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1, 3 \rangle$

► $s = \langle 25, 9 \rangle$, $resultado = \langle 5, 3 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- e) problema `raicesCuadradasDos` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
}

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- e) problema `raicesCuadradasDos` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
}

► $s = \langle 4, 1, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- e) problema `raicesCuadradasDos` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 1, 2, 3 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- e) problema `raicesCuadradasDos` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
}

- ▶ $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 1, 2, 3 \rangle$
- ▶ $s = \langle 25, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- e) problema `raicesCuadradasDos` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 requiere: {No hay elementos repetidos en s }
 asegura: {*resultado* tiene la misma cantidad de
 elementos que s }
 asegura: {Los elementos de *resultado* son la salida de
 aplicar el problema `raizCuadrada()` a todos los
 elementos de la secuencia s }
}

- ▶ $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 1, 2, 3 \rangle$
- ▶ $s = \langle 25, 9 \rangle$, $resultado = \langle 3, 5 \rangle$

Guía 2 - Ejercicio 2

2. ¿Qué consecuencia tiene la diferencia de asegura en el resultado entre los problemas `raicesCuadradasUno` y `raicesCuadradasDos`?

```
problema raicesCuadradasUno (s: seq<Z>) : seq<R> {  
  requiere: {...}  
  asegura: {resultado tiene la misma cantidad de elementos que s}  
  asegura: {Los elementos de resultado son la salida de aplicar el problema  
            raizCuadrada() a todos los elem de la secuencia s}  
  asegura: {El orden de la secuencia resultado es el mismo que en la secuencia s}  
}
```

```
problema raicesCuadradasDos (s: seq<Z>) : seq<R> {  
  requiere: {...}  
  asegura: {resultado tiene la misma cantidad de elementos que s}  
  asegura: {Los elementos de resultado son la salida de aplicar el problema  
            raizCuadrada() a todos los elem de la secuencia s}  
}
```

Guía 2 - Ejercicio 2

2. ¿Qué consecuencia tiene la diferencia de asegura en el resultado entre los problemas raicesCuadradasUno y raicesCuadradasDos?

```
problema raicesCuadradasUno (s: seq<Z>) : seq<R> {  
  requiere: {...}  
  asegura: {resultado tiene la misma cantidad de elementos que s}  
  asegura: {Los elementos de resultado son la salida de aplicar el problema  
            raizCuadrada() a todos los elem de la secuencia s}  
  asegura: {El orden de la secuencia resultado es el mismo que en la secuencia s}  
}
```

```
problema raicesCuadradasDos (s: seq<Z>) : seq<R> {  
  requiere: {...}  
  asegura: {resultado tiene la misma cantidad de elementos que s}  
  asegura: {Los elementos de resultado son la salida de aplicar el problema  
            raizCuadrada() a todos los elem de la secuencia s}  
}
```

3. En base a la respuesta del ítem anterior, ¿un algoritmo que satisface la especificación de raicesCuadradasUno, también satisface la especificación de raicesCuadradasDos? ¿y al revés?

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

► $s = \langle 4, 1, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

► $s = \langle 4, 1, 9 \rangle$, *resultado* = $\langle 2, 1, 3, 0, 0 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

► $s = \langle 4, 1, 9 \rangle$, *resultado* = $\langle 2, 1, 3, 0, 0 \rangle$

► $s = \langle 25, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

► $s = \langle 4, 1, 9 \rangle$, *resultado* = $\langle 2, 1, 3, 0, 0 \rangle$

► $s = \langle 25, 9 \rangle$, *resultado* = $\langle 5, 3 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

► $s = \langle 4, 1, 9 \rangle$, *resultado* = $\langle 2, 1, 3, 0, 0 \rangle$

► $s = \langle 25, 9 \rangle$, *resultado* = $\langle 5, 3 \rangle$

► $s = \langle 1 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

h) problema `raicesCuadradasCinco` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s (si esa posición existe
 en s)}}
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1, 3, 0, 0 \rangle$

► $s = \langle 25, 9 \rangle$, $resultado = \langle 5, 3 \rangle$

► $s = \langle 1 \rangle$, $resultado = \langle 1, 1, 1, 1 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{R}\rangle$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la
 misma que s }
 asegura: {Cada posición de *resultado* es la salida de
 aplicar `raizCuadrada()` a cada elemento que se
 encuentra en esa posición en s }
}

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq(\mathbb{Z})$) : $seq(\mathbb{R})$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la misma que s }
 asegura: {Cada posición de *resultado* es la salida de aplicar `raizCuadrada()` a cada elemento que se encuentra en esa posición en s }
}

► $s = \langle 4, 1, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq(\mathbb{Z})$) : $seq(\mathbb{R})$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la misma que s }
 asegura: {Cada posición de *resultado* es la salida de aplicar `raizCuadrada()` a cada elemento que se encuentra en esa posición en s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq(\mathbb{Z})$) : $seq(\mathbb{R})$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la misma que s }
 asegura: {Cada posición de *resultado* es la salida de aplicar `raizCuadrada()` a cada elemento que se encuentra en esa posición en s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1 \rangle$

► $s = \langle 25, 9 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq(\mathbb{Z})$) : $seq(\mathbb{R})$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la misma que s }
 asegura: {Cada posición de *resultado* es la salida de aplicar `raizCuadrada()` a cada elemento que se encuentra en esa posición en s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1 \rangle$

► $s = \langle 25, 9 \rangle$, $resultado = \langle 5, 3 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq(\mathbb{Z})$) : $seq(\mathbb{R})$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la misma que s }
 asegura: {Cada posición de *resultado* es la salida de aplicar `raizCuadrada()` a cada elemento que se encuentra en esa posición en s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1 \rangle$

► $s = \langle 25, 9 \rangle$, $resultado = \langle 5, 3 \rangle$

► $s = \langle 1, 2, 3, 4, 5, 6 \rangle$

Guía 2 - Ejercicio 1

Dadas las siguientes especificaciones, dar valores de entrada y salida que cumplan con los requiere y asegura respectivamente:

- i) problema `raicesCuadradasSeis` ($s: seq(\mathbb{Z})$) : $seq(\mathbb{R})$ {
 requiere: {Todos los elementos de s son positivos}
 asegura: {La longitud de *resultado* es como máximo la misma que s }
 asegura: {Cada posición de *resultado* es la salida de aplicar `raizCuadrada()` a cada elemento que se encuentra en esa posición en s }
}

► $s = \langle 4, 1, 9 \rangle$, $resultado = \langle 2, 1 \rangle$

► $s = \langle 25, 9 \rangle$, $resultado = \langle 5, 3 \rangle$

► $s = \langle 1, 2, 3, 4, 5, 6 \rangle$, $resultado = \langle 1 \rangle$

Guía 2 - Ejercicio 2

4. Explicar en palabras las diferencias entre los problemas `raicesCuadradasCinco` y `raicesCuadradasSeis`. ¿Cómo influye el `asegura` de longitud máxima?

```
problema raicesCuadradasCinco (s: seq<ℤ>) : seq<ℝ> {  
  requiere: {Todos los elementos de s son positivos}  
  asegura: {Cada posición de resultado es la salida de aplicar raizCuadrada() a cada  
            elemento que se encuentra en esa posición en s}  
}
```

```
problema raicesCuadradasSeis (s: seq<ℤ>) : seq<ℝ> {  
  requiere: {Todos los elementos de s son positivos}  
  asegura: {La longitud de resultado es como máximo la misma que s}  
  asegura: {Cada posición de resultado es la salida de aplicar raizCuadrada() a cada  
            elemento que se encuentra en esa posición en s}  
}
```

Guía 2 - Ejercicio 2

7. ¿Qué ocurre si eliminamos los requiere “no hay repetidos” ? ¿Es $\langle 2, 2, 1 \rangle$ una salida válida para el problema raícesCuadradasDos dado $s = \langle 4, 1, 1 \rangle$?

Guía 2 - Ejercicio 3

Responder las preguntas dada la siguiente especificación para el problema de ordenar una secuencia de enteros (es decir, dada una secuencia de enteros, devolver los mismos elementos ordenados de menor a mayor):

```
problema ordenar (s:  $seq\langle\mathbb{Z}\rangle$ ) :  $seq\langle\mathbb{Z}\rangle$  {  
  requiere: {True}  
  asegura: {resultado es una secuencia en la cual cada elemento  
            es estrictamente mayor que el anterior}  
}
```

Guía 2 - Ejercicio 3

Responder las preguntas dada la siguiente especificación para el problema de ordenar una secuencia de enteros (es decir, dada una secuencia de enteros, devolver los mismos elementos ordenados de menor a mayor):

```
problema ordenar (s: seq<Z>) : seq<Z> {  
  requiere: {True}  
  asegura: {resultado es una secuencia en la cual cada elemento  
            es estrictamente mayor que el anterior}  
}
```

- a) Dado $s = \langle 4, 3, 5 \rangle$ como secuencia de entrada, ¿es resultado $= \langle 3, 4, 5 \rangle$ una solución válida según la especificación?

Guía 2 - Ejercicio 3

Responder las preguntas dada la siguiente especificación para el problema de ordenar una secuencia de enteros (es decir, dada una secuencia de enteros, devolver los mismos elementos ordenados de menor a mayor):

problema ordenar ($s: seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$ {
 requiere: {True}
 asegura: {*resultado* es una secuencia en la cual cada elemento
 es estrictamente mayor que el anterior}
}

- b) Dado $s = \langle 4, 3, 3, 5 \rangle$ como secuencia de entrada, ¿es $resultado = \langle 3, 3, 4, 5 \rangle$ una solución válida según la especificación? Corregir la especificación modificando el *requiere*.

Guía 2 - Ejercicio 3

Responder las preguntas dada la siguiente especificación para el problema de ordenar una secuencia de enteros (es decir, dada una secuencia de enteros, devolver los mismos elementos ordenados de menor a mayor):

```
problema ordenar (s: seq<ℤ>) : seq<ℤ> {  
  requiere: {True}  
  asegura: {resultado es una secuencia en la cual cada elemento  
            es estrictamente mayor que el anterior}  
}
```

- c) Si tomamos $s = \langle 4, 3, 5 \rangle$ como secuencia de entrada, ¿es $resultado = \langle 3, 4 \rangle$ una solución válida según la especificación?
Corregir la especificación modificando el *asegura*

Guía 2 - Ejercicio 3

Responder las preguntas dada la siguiente especificación para el problema de ordenar una secuencia de enteros (es decir, dada una secuencia de enteros, devolver los mismos elementos ordenados de menor a mayor):

```
problema ordenar (s: seq<ℤ>) : seq<ℤ> {  
  requiere: {True}  
  asegura: {resultado es una secuencia en la cual cada elemento  
            es estrictamente mayor que el anterior}  
}
```

- d) Si tomamos $s = \langle 4, 3, 5 \rangle$ como secuencia de entrada, ¿es $resultado = \langle 3, 4 \rangle$ una solución válida según la especificación?
Corregir la especificación modificando el *asegura*

Guía 2 - Ejercicio 3

Responder las preguntas dada la siguiente especificación para el problema de ordenar una secuencia de enteros (es decir, dada una secuencia de enteros, devolver los mismos elementos ordenados de menor a mayor):

```
problema ordenar (s: seq<ℤ>) : seq<ℤ> {  
  requiere: {True}  
  asegura: {resultado es una secuencia en la cual cada elemento  
            es estrictamente mayor que el anterior}  
}
```

- e) Escribir una especificación que permita recibir cualquier secuencia s como parámetro y garantice que *resultado* contiene el resultado de ordenar correctamente s .

Guía 2 - Ejercicio 5

A Ciudad Universitaria (CU) llegan 8 líneas de colectivos (28, 33, 34, 37, 45, 107, 160, 166). Con el fin de controlar la frecuencia diaria de cada una de ellas, un grupo de investigación del Departamento de Computación instaló cámaras y sistemas de reconocimiento de imágenes en el ingreso al predio. Durante cada día dicho sistema identifica y registra cada colectivo que entra, almacenando la información de a qué línea pertenece en una secuencia.

Guía 2 - Ejercicio 5

A Ciudad Universitaria (CU) llegan 8 líneas de colectivos (28, 33, 34, 37, 45, 107, 160, 166). Con el fin de controlar la frecuencia diaria de cada una de ellas, un grupo de investigación del Departamento de Computación instaló cámaras y sistemas de reconocimiento de imágenes en el ingreso al predio. Durante cada día dicho sistema identifica y registra cada colectivo que entra, almacenando la información de a qué línea pertenece en una secuencia.

- a) Especificar el problema `cantidadColectivosLinea()` que a partir del número de una de las líneas que entra a CU y una lista que cumpla con la descripción del sistema presentado devuelva cuántos colectivos de esa línea ingresaron durante el día.

Guía 2 - Ejercicio 5

- a) Especificar el problema `cantColectivosLinea()` que a partir del número de una de las líneas que entra a CU y una lista que cumpla con la descripción del sistema presentado devuelva cuántos colectivos de esa línea ingresaron durante el día.

problema `cantColectivosLinea (linea: \mathbb{Z} , bondis: $seq\langle\mathbb{Z}\rangle$) : \mathbb{Z} {`

 requiere: {Todos los elementos de la secuencia *bondis*
 pertenecen a (28, 33, 34, 37, 45, 107, 160, 166)}

 requiere: {*linea* es alguno de los siguientes números: 28, 33,
 34, 37, 45, 107, 160, 166.}

 asegura: {*resultado* es la cantidad de veces que *linea* aparece
 en la secuencia *bondis*}

}

Guía 2 - Ejercicio 5

- b) Especificar el problema `compararLineas()` que a partir de los números de 2 líneas y de una lista que cumpla con la descripción del sistema presentado devuelva cuál de las dos líneas tiene mejor frecuencia diaria (utilizar `cantColectivosLinea()`)

Guía 2 - Ejercicio 5

- b) Especificar el problema `compararLineas()` que a partir de los números de 2 líneas y de una lista que cumpla con la descripción del sistema presentado devuelva cuál de las dos líneas tiene mejor frecuencia diaria (utilizar `cantColectivosLinea()`)

```
problema compararLineas (l1:  $\mathbb{Z}$ , l2:  $\mathbb{Z}$ , bondis:  $seq\langle\mathbb{Z}\rangle$ ) :  $\mathbb{Z}$  {  
  requiere: {Todos los elementos de la secuencia bondis pertenecen a  
    (28, 33, 34, 37, 45, 107, 160, 166)}  
  requiere: {l1 es alguno de los siguientes números: 28, 33, 34, 37,  
    45, 107, 160, 166.}  
  requiere: {l2 es alguno de los siguientes números: 28, 33, 34, 37,  
    45, 107, 160, 166.}  
  asegura: {Si cantColectivosLinea(l1, bondis) >  
    cantColectivosLinea(l2, bondis), entonces resultado = l1}  
  asegura: {Si cantColectivosLinea(l2, bondis) >  
    cantColectivosLinea(l1, bondis), entonces resultado = l2}  
}
```


Guía 2 - Ejercicio 5

- b) Especificar el problema `compararLineas()` que a partir de los números de 2 líneas y de una lista que cumpla con la descripción del sistema presentado devuelva cuál de las dos líneas tiene mejor frecuencia diaria (utilizar `cantColectivosLinea()`)

```
problema compararLineas (l1:  $\mathbb{Z}$ , l2:  $\mathbb{Z}$ , bondis:  $seq\langle\mathbb{Z}\rangle$ ) :  $\mathbb{Z}$  {  
  requiere: {Todos los elementos de la secuencia bondis pertenecen a  
    (28, 33, 34, 37, 45, 107, 160, 166)}  
  requiere: {l1 es alguno de los siguientes números: 28, 33, 34, 37,  
    45, 107, 160, 166.}  
  requiere: {l2 es alguno de los siguientes números: 28, 33, 34, 37,  
    45, 107, 160, 166.}  
  asegura: {Si cantColectivosLinea(l1, bondis) >  
    cantColectivosLinea(l2, bondis), entonces resultado = l1}  
  asegura: {Si cantColectivosLinea(l2, bondis) >  
    cantColectivosLinea(l1, bondis), entonces resultado = l2}  
}
```

¿Qué debería ocurrir si

cantColectivosLinea(l1, *bondis*) = *cantColectivosLinea*(l2, *bondis*)?

Guía 2 - Ejercicio 5

- b) Especificar el problema `compararLineas()` que a partir de los números de 2 líneas y de una lista que cumpla con la descripción del sistema presentado devuelva cuál de las dos líneas tiene mejor frecuencia diaria (utilizar `cantColectivosLinea()`)

problema `compararLineas (l1: \mathbb{Z} , l2: \mathbb{Z} , bondis: $seq\langle\mathbb{Z}\rangle$) : \mathbb{Z} {`

 requiere: {Todos los elementos de la secuencia *bondis* pertenecen a (28, 33, 34, 37, 45, 107, 160, 166)}

 requiere: {l1 es alguno de los siguientes números: 28, 33, 34, 37, 45, 107, 160, 166.}

 requiere: {l2 es alguno de los siguientes números: 28, 33, 34, 37, 45, 107, 160, 166.}

 asegura: { $cantColectivosLinea(result, bondis) \geq cantColectivosLinea(l1, bondis) \wedge cantColectivosLinea(result, bondis) \geq cantColectivosLinea(l2, bondis)$ }

}

Guía 2 - Ejercicio 5

- b) Especificar el problema `compararLineas()` que a partir de los números de 2 líneas y de una lista que cumpla con la descripción del sistema presentado devuelva cuál de las dos líneas tiene mejor frecuencia diaria (utilizar `cantColectivosLinea()`)

problema `compararLineas (l1: \mathbb{Z} , l2: \mathbb{Z} , bondis: $seq\langle\mathbb{Z}\rangle$) : \mathbb{Z} {`

 requiere: {Todos los elementos de la secuencia *bondis* pertenecen a (28, 33, 34, 37, 45, 107, 160, 166)}

 requiere: {l1 es alguno de los siguientes números: 28, 33, 34, 37, 45, 107, 160, 166.}

 requiere: {l2 es alguno de los siguientes números: 28, 33, 34, 37, 45, 107, 160, 166.}

 asegura: { $cantColectivosLinea(result, bondis) \geq cantColectivosLinea(l1, bondis) \wedge cantColectivosLinea(result, bondis) \geq cantColectivosLinea(l2, bondis)$ }

}

¿Podemos evaluar *cantColectivosLinea(result, bondis)* sin pedir nada sobre result?

Guía 2 - Ejercicio 5

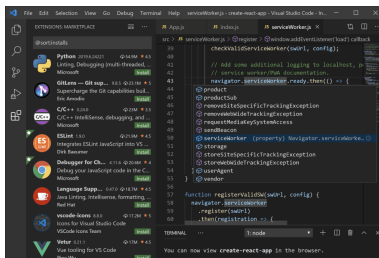
- b) Especificar el problema `compararLineas()` que a partir de los números de 2 líneas y de una lista que cumpla con la descripción del sistema presentado devuelva cuál de las dos líneas tiene mejor frecuencia diaria (utilizar `cantColectivosLinea()`)

```
problema compararLineas (l1:  $\mathbb{Z}$ , l2:  $\mathbb{Z}$ , bondis:  $seq\langle\mathbb{Z}\rangle$ ) :  $\mathbb{Z}$  {  
  requiere: {Todos los elementos de la secuencia bondis pertenecen a  
    (28, 33, 34, 37, 45, 107, 160, 166)}  
  requiere: {l1 es alguno de los siguientes números: 28, 33, 34, 37,  
    45, 107, 160, 166.}  
  requiere: {l2 es alguno de los siguientes números: 28, 33, 34, 37,  
    45, 107, 160, 166.}  
  asegura: {result = l1  $\vee$  result = l2}  
  asegura: {cantColectivosLinea(result, bondis)  $\geq$   
    cantColectivosLinea(l1, bondis)  $\wedge$   
    cantColectivosLinea(result, bondis)  $\geq$   
    cantColectivosLinea(l2, bondis)}  
}
```

Configurando Vscode

VS code es una IDE (Integrated Development Environment), existen MUCHAS:

- ▶ Visual Studio (<https://visualstudio.microsoft.com/es/>)
- ▶ Eclipse (<https://www.eclipse.org/>)
- ▶ IntelliJ IDEA (<https://www.jetbrains.com/es-es/idea/>)
- ▶ **Visual Code o Visual Studio Code** (<https://code.visualstudio.com/>)
 - ▶ Es un editor de textos que se “convierte” en IDE mediante *extensions*.
 - ▶ Lo utilizaremos para programar en Haskell y Python.



Configurando Vscode

Vamos a instalar la extensión de Haskell:

- ▶ Abrir Visual Studio Code en sus computadoras

Configurando Vscode

Vamos a instalar la extensión de Haskell:

- ▶ Abrir Visual Studio Code en sus computadoras
- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)

Configurando Vscode

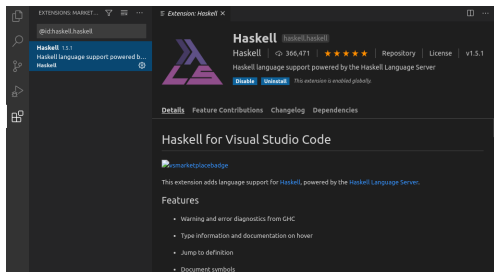
Vamos a instalar la extensión de Haskell:

- ▶ Abrir Visual Studio Code en sus computadoras
- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)
- ▶ Buscar `ext install haskell.haskell`

Configurando Vscode

Vamos a instalar la extensión de Haskell:

- ▶ Abrir Visual Studio Code en sus computadoras
- ▶ Abrir el buscador apretando `ctrl+P` (se abre una barra arriba)
- ▶ Buscar `ext install haskell.haskell`
- ▶ En la barra de la izquierda se abre el buscador de extensiones con una sola opción encontrada. Hacemos click y la instalamos (si no lo está).



Configurando Vscode

Ahora la extensión de Syntax Highlighting: (si no funciona no es tan grave)

- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)

Configurando Vscode

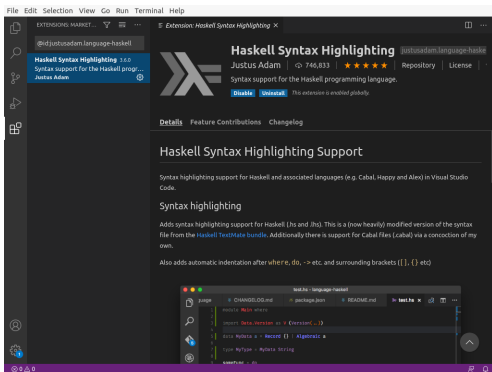
Ahora la extensión de Syntax Highlighting: (si no funciona no es tan grave)

- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)
- ▶ Buscar `ext install justusadam.language-haskell`

Configurando Vscode

Ahora la extensión de Syntax Highlighting: (si no funciona no es tan grave)

- ▶ Abrir el buscador apretando `ctrl+P` (se abre una barra arriba)
- ▶ Buscar `ext install justusadam.language-haskell`
- ▶ En la barra de la izquierda se abre el buscador de extensiones con una sola opción encontrada. Hacemos click y la instalamos (si no lo está).



Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en Escritorio/guia3/
- ▶ Abrir una Terminal Terminal > New Terminal

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

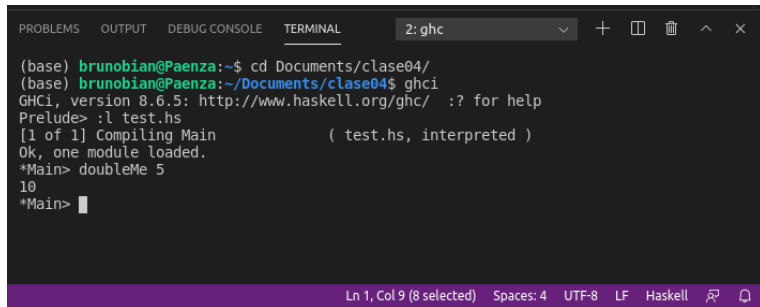
- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`
- ▶ Dentro del intérprete tenemos que pedirle que cargue nuestro archivo: `:l test.hs`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función?
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`
- ▶ Dentro del intérprete tenemos que pedirle que cargue nuestro archivo: `:l test.hs`
- ▶ Ahora nuestra función ya existe y podemos usarla `doubleMe 5`

Primeros pasos en Haskell



A screenshot of a terminal window with a dark background. The window has tabs at the top: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, and a dropdown menu shows '2: ghc'. The terminal text shows a user running 'cd Documents/clase04/' and 'ghci'. It then shows 'GHCi, version 8.6.5: http://www.haskell.org/ghc/ :? for help'. The user enters ':l test.hs', and the terminal shows '[1 of 1] Compiling Main (test.hs, interpreted)' and 'Ok, one module loaded.'. The user then enters '*Main> doubleMe 5', and the terminal shows '10'. The bottom status bar is purple and shows 'Ln 1, Col 9 (8 selected) Spaces: 4 UTF-8 LF Haskell' along with some icons.

```
(base) brunobian@Paenza:~$ cd Documents/clase04/
(base) brunobian@Paenza:~/Documents/clase04$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/ :? for help
Prelude> :l test.hs
[1 of 1] Compiling Main                ( test.hs, interpreted )
Ok, one module loaded.
*Main> doubleMe 5
10
*Main> 
```

Ln 1, Col 9 (8 selected) Spaces: 4 UTF-8 LF Haskell

Ya tenemos todo lo necesario para hacer la Guía 3
Ahora a programar!!

Guía 3 - Ejercicio 1

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

$$f(1) = 8, f(4) = 131, f(16) = 16$$

cuya especificación es la siguiente:

```
problema f (n:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere:  $\{n = 1 \vee n = 4 \vee n = 16\}$   
  asegura:  $\{(n = 1 \rightarrow result = 8) \wedge (n = 4 \rightarrow result = 131) \wedge (n =$   
            $16 \rightarrow result = 16)\}$   
}
```

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

$$g(8) = 16, g(16) = 4, g(131) = 1$$

- c) A partir de las funciones definidas en los ítems 1 y 2, implementar las funciones parciales $h = f \circ g$ y $k = g \circ f$

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- c) **maximo3**: devuelve el máximo entre tres números enteros.

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Una especificación semi-formal

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { res es igual a  $x$ , o a  $y$  o a  $z$  }  
  asegura: { res es mayor o igual a  $x$ , y a  $y$ , y a  $z$  }  
}
```

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Una especificación semi-formal

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { res es igual a  $x$ , o a  $y$  o a  $z$  }  
  asegura: { res es mayor o igual a  $x$ , y a  $y$ , y a  $z$  }  
}
```

Usando lógica...

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Una especificación semi-formal

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { res es igual a  $x$ , o a  $y$  o a  $z$  }  
  asegura: { res es mayor o igual a  $x$ , y a  $y$ , y a  $z$  }  
}
```

Usando lógica...

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { ( $res = x$ )  $\vee$  ( $res = y$ )  $\vee$  ( $res = z$ ) }  
  asegura: { ( $res \geq x$ )  $\wedge$  ( $res \geq y$ )  $\wedge$  ( $res \geq z$ ) }  
}
```

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Esto tiene (al menos) dos interpretaciones posibles:

- ▶ Cuando hay algún número repetido no lo sumo
 - ▶ $\text{sumaDistintos}(1,1,2) = 2$
- ▶ Cuando hay algún número repetido lo sumo una sola vez
 - ▶ $\text{sumaDistintos}(1,1,2) = 3$

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Esto tiene (al menos) dos interpretaciones posibles:

- ▶ Cuando hay algún número repetido no lo sumo
 - ▶ $\text{sumaDistintos}(1,1,2) = 2$
- ▶ Cuando hay algún número repetido lo sumo una sola vez
 - ▶ $\text{sumaDistintos}(1,1,2) = 3$

Una especificación **semi-formal** de la primera opción

```
problema sumaDistintos (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: { - }  
  asegura: {si los 3 parámetros son distintos entonces  $res = x + y + z$ }  
  asegura: {si 2 parámetros son iguales, res es igual al no repetido}  
  asegura: {si los 3 parámetros son iguales,  $res = 0$ }  
}
```

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Esto tiene (al menos) dos interpretaciones posibles:

- ▶ Cuando hay algún número repetido no lo sumo
 - ▶ $\text{sumaDistintos}(1,1,2) = 2$
- ▶ Cuando hay algún número repetido lo sumo una sola vez
 - ▶ $\text{sumaDistintos}(1,1,2) = 3$

Una especificación **formal** de la primera opción

```
problema sumaDistintos (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: {(  $x \neq y$ )  $\wedge$  ( $x \neq z$ )  $\wedge$  ( $y \neq z$ ) }  $\rightarrow res = x + y + z$   
  asegura: {(  $x = y$ )  $\wedge$  ( $x \neq z$ )  $\wedge$  ( $y \neq z$ ) }  $\rightarrow res = z$   
  asegura: {(  $x \neq y$ )  $\wedge$  ( $x = z$ )  $\wedge$  ( $y \neq z$ ) }  $\rightarrow res = y$   
  asegura: {(  $x \neq y$ )  $\wedge$  ( $x \neq z$ )  $\wedge$  ( $y = z$ ) }  $\rightarrow res = x$   
  asegura: {(  $x = y$ )  $\wedge$  ( $x = z$ )  $\wedge$  ( $y = z$ ) }  $\rightarrow res = 0$   
}
```

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- i) **digitoUnidades:** dado un número natural, extrae su dígito de las unidades.

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- i) **digitoUnidades**: dado un número natural, extrae su dígito de las unidades.

```
problema digitoUnidades (x:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { result es el último dígito de x }  
}
```

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- j) **digitoUnidades:** dado un número natural, extrae su dígito de las decenas.

Guía 3 - Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

j) **digitoUnidades**: dado un número natural, extrae su dígito de las decenas.

```
problema digitoDecenas (x:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { result es el dígito de x correspondiente a las  
             decenas}  
}
```

Guía 3 - 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.

Guía 3 - 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 }  
}
```