

FINAL DE PARADIGMAS DE PROGRAMACION

Legajo.: _____

Alumno: _____

Punto 1

Considerando los siguientes ejemplos de código en Haskell y Smalltalk

Grupo 1:	Grupo 2:
<pre>S1: Smalltalk alquileresVencidos (Cliente) "alquileres es una colección de alquileres" ^ alquileres select:[:alquiler alquiler estaVencido]</pre>	<pre>S2: Smalltalk alquileresVencidos (Cliente) "alquileres es una colección de alquileres" alqVencidos alqVencidos := Set new. alquileres do: [:alq alq estaVencido ifTrue:[alqVencidos add: alq].]. ^alqVencidos</pre>
<pre>H1: Haskell filtrarPares (x:xs) = filter even (x:xs)</pre>	<pre>H2: Haskell filtrarPares [] = [] filtrarPares (x:xs) even x = x:filtrarPares xs otherwise = filtrarPares xs</pre>

1. ¿Qué grupo de ejemplos elegiría y por qué? ¿Que paralelismo encuentra entre los dos grupos de ejemplos?
2. Si al ejemplo H1 y al ejemplo S1 se le agrega un **head** adelante y un **first** atrás, respectivamente: si la colección/lista es grande, ¿hay problemas de performance? ¿En H, en S o en ninguno de ellos?
3. Desarrollar las principales diferencias entre paradigmas o esquemas declarativos y procedurales.

Punto 2

Dada la siguiente función:

```
func [ ] xs _ = xs
func xs [ ] _ = xs
func (x:xs) (y:ys) z = z x y : func xs ys z
```

1. ¿Qué errores tiene el siguiente dominio e imagen propuesto para la función? Corregir y justificar.

```
func :: [a] -> [b] -> (a-> b) -> [b]
```

2. Considerando este ejemplo.
 - a) Además de la recursividad, que otro concepto se está utilizando?
 - b) Provea un ejemplo de uso de la función *func*, comentando la aplicación del concepto mencionado.
 - c) ¿Cuales son las ventajas de usar este concepto?

FINAL DE PARADIGMAS DE PROGRAMACION

Punto 3

Considerando estas dos opciones de código fuente Smalltalk:

<p>Opción 1</p> <pre>saldo (Cliente) total total := 0. self facturas do: [:fact fact items do: [:item total + item cantidad * item producto precioUnitario]]. ^total</pre>	<div style="display: flex; justify-content: space-around;"><div style="border: 1px solid black; padding: 5px; width: 45%;"><p style="text-align: center;">Cliente</p><hr/><p style="text-align: center;">facturas</p><hr/></div><div style="border: 1px solid black; padding: 5px; width: 45%;"><p style="text-align: center;">Factura</p><hr/><p style="text-align: center;">items</p><hr/></div></div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"><div style="border: 1px solid black; padding: 5px; width: 45%;"><p style="text-align: center;">Item</p><hr/><p style="text-align: center;">producto cantidad</p><hr/></div><div style="border: 1px solid black; padding: 5px; width: 45%;"><p style="text-align: center;">Producto</p><hr/><p style="text-align: center;">nombre precioUnitario</p><hr/></div></div>
<p>Opción 2</p> <pre>saldo (Cliente) ^self facturas inject: 0 into: [:tot :fact tot + fact importe] importe (Factura) ^self items inject: 0 into: [:tot :item tot + item importe] importe (Item) ^self cantidad * self producto precioUnitario</pre>	

1. ¿Cuál de las dos le parece mejor y con que concepto esta relacionada esta elección?
2. En caso de incorporarse un nuevo tipo de documento que no tiene ítems. ¿Qué hay que modificar/agregar en los modelos anteriores?.
3. ¿Cómo se refleja la ventaja de un modelo sobre el otro respecto de este agregado?.

Punto 4

1. ¿Cómo se relaciona el concepto de declaratividad con el motor de inferencia de Prolog?
2. ¿Cuáles son las diferencias entre el concepto de **functor** / **tupla** y el de **lista** respecto al concepto de tipo?