FINAL DE PARADIGMAS DE PROGRAMACION

Legajo: _			
Alumno: _	 	 	

Punto 1

Considerando los siguientes ejemplos de código en Haskell y Smalltalk

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

- 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 perfomance? ¿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.

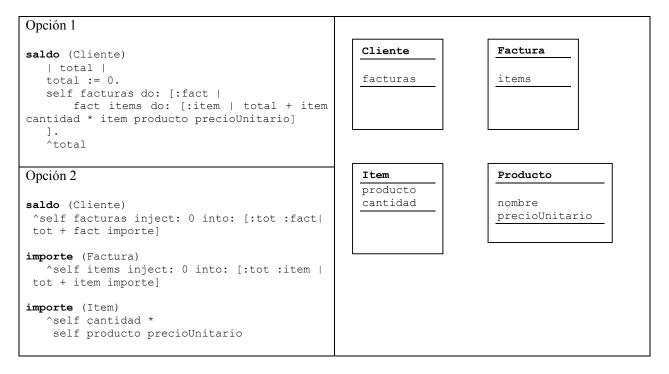
```
func :: [a] \rightarrow [b] \rightarrow (a \rightarrow b) \rightarrow [b]
```

- 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:



- 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?