FINAL DE PARADIGMAS DE PROGRAMACION
FECHA: 13-02-2010

Punto 1
a. Tenemos estas dos soluciones para saber la longitud de un conjunto de datos:

PROLOG	**Haskell**
longitud([], 0). longitud([_|Xs], Longitud):- longitud(Xs, LongitudXs), Longitud is LongitudXs + 1.	longitud [] = 0 longitud (x:xs) = 1 + longitud xs

Justificar por qué ambas soluciones son polimórficas.

b. Se quiere evaluar las siguientes consultas:

PROLOG	**Haskell**
? longitud([8, hermanos])	Main>longitud [8, hermanos]

¿Qué ocurre en cada caso? ¿Con qué concepto está relacionado?

c. Se codificó una tercera solución en Smalltalk, definiendo un método **#size** que curiosamente también es polimórfico. ¿Por qué afirmamos que lo es? Dé ejemplos de uso donde se demuestre esta cualidad.

Punto 2
Para definir el factorial de un número tenemos estas dos soluciones:

PROLOG	**Haskell**
factorial(0, 1). factorial(N,FactorialN):- N1 is N – 1, factorial(N1, FactorialN1), FactorialN is N * FactorialN1.	factorial n | n == 0 = 1 | n > 0 = n * factorial (n – 1)

a. ¿En qué caso estamos en presencia de pattern matching? ¿Cuál es el concepto que se utiliza en la otra solución?
b. Cuando se pide el factorial de un número negativo, indique en qué solución se produce un error, en cuál un loop infinito, y justifique por qué ocurre.
c. Si tuviera que implementar el cálculo de factorial en Objetos, ¿cuál sería el objeto receptor y cuál el método?

Punto 3
Un museo colecciona obras de arte, para conocer las obras valiosas se tiene el siguiente código:

Smalltalk
```
>>Museo (VI: obras)
obrasValiosas
   ^obras select: [ :obra | obra esValiosa ]

>>Obra (VI: tipo)
esValiosa
    obra tipo = 'Pintura' ifTrue: [ ^self esValiosaPintura ]
    obra tipo = 'Escultura' ifTrue: [ ^self esValiosaEscultura ]
    obra tipo = 'Arquitectura' ifTrue: [ ^self esValiosaArquitectura ]
```

```
esValiosaEscultura
    ^false
esValiosaArquitectura
    ^false
esValiosaPintura
    ^false

>>Pintura (VI: anio autor)
esValiosaPintura
   ^ anio < 1800 or: [ autor esReconocido]
…
 (Análogamente  con el resto de los métodos, que se redefinen en cada subclase)
```

a. ¿Cuál es el error conceptual de la solución anterior? Justifique con los conceptos que apliquen al contexto.
b. Indique qué modificaría del código anterior y qué conceptos aprovecha con la nueva solución.
c. Uno de los desarrolladores pensó otra solución para conocer las obras valiosas:

Smalltalk

```
>>Museo (VI: obras, obrasValiosas)
generarObrasValiosas
   obrasValiosas := obras select: [ :obra | obra esValiosa ]

obrasValiosas
   ^obrasValiosas
```

d. La nueva solución introduce un concepto que antes no aparecía, ¿cuál es ese concepto?
e. ¿Qué desventajas presenta la aparición de ese concepto en la solución? Justifique con un ejemplo concreto.

Punto 4

Se tiene la siguiente función en Haskell:

```
filter          :: (a -> Bool) -> [a] -> [a]
filter f xs       = [ x | x <- xs, f x ]
```

a. ¿Cuál es la ventaja de que filter sea una función de orden superior? O sea, ¿qué ocurriría con la solución si no existiera ese concepto?
b. Compare esa solución con respecto a la siguiente

```
filter f []      = []
filter f (x:xs) | f x              = x : filter f xs
                | otherwise = filter f xs
```

¿Cuál de las dos soluciones tiene más características declarativas? Justifique su decisión.