

1 - Dadas las siguientes definiciones dadas en los lenguajes Haskell y Prolog

a) la función **concatenar**

concatenar [] ys = ys

concatenar (x:xs) ys = x:concatenar xs ys

b) el predicado **concatenar/3**, definido por las siguientes cláusulas

concatenar([], Ys, Ys).

concatenar([X|Xs], Ys, [X|Zs]):-concatenar(Xs, Ys, Zs).

Describa cómo se aplican, en cada item, los conceptos de

- inversibilidad
- pattern matching
- tipos de datos genéricos/polimorfismo

2 – Para un sistema de sueldos se tiene el siguiente método:

>>Empresa

actualizarSueldosSegun: *unPorcentaje*

```
empleados do: [ :empleado | empleado sueldo: (empleado sueldo +  
empleado sueldo * unPorcentaje) ]
```

- a) ¿De qué manera podría mejorar el código planteado? Si no se le ocurre nada, lea el punto 'd' antes de continuar. ¿Con qué concepto se relaciona la mejora? Explique y codifique.
- b) Traslade la misma solución a Haskell.
- c) ¿Qué problemas encontró para manejar el estado de los empleados? ¿Cómo lo resolvió? Justifique relacionando con los conceptos que apliquen a la pregunta.

Ahora se agrega como requerimiento contemplar dos tipos nuevos de empleado:

- el comisionista, que no trabaja con sueldo fijo sino con el porcentaje de las ventas
- el empleado jerárquico, que tiene empleados a cargo.

Al actualizar los sueldos:

- al comisionista no hay que actualizarle nada (ya que no puede aumentar el % de comisión de la venta)
 - si un empleado jerárquico tiene más de 5 empleados a cargo, se le aumenta un 5%, en caso contrario se le aumenta un 3%
- d) Codifique el nuevo requerimiento en Objetos.
 - e) ¿Qué concepto saliente aparece en su solución? ¿Quién es el que se beneficia con la aparición de este concepto?

- f) ¿Es posible trasladar la misma solución a Haskell? Justifique por qué/por qué no y con qué concepto está relacionado (sólo considerar los conceptos que apliquen a la pregunta).

Aparece un nuevo requerimiento: se registra la fecha de ingreso de todos los empleados de manera de poder responder la antigüedad de cualquiera de los empleados, sean comisionistas, empleados comunes o jerárquicos.

- g) ¿Qué modificaciones haría en su solución? Indique en qué clase ubicaría dichos cambios (no hace falta codificar).
h) ¿Qué concepto permite evitar repetición de código? Justifique.

3- Para el siguiente programa Prolog

```
partido(lanus, 3, quilmes, 1).  
partido(lanus, 2, niuls, 4).  
partido(central, 1, velez, 0).  
partido(lanus, 1, central, 2).  
partido(velez, 1, quilmes, 2).  
partido(lanus, 1, banfield, 2).  
  
leGano(E1, E2):- partido(E1, G1, E2, G2), G1 > G2.  
leGano(E1, E2):- partido(E2, G2, E1, G1), G1 > G2.  
  
esMuyCapo(E):- forall(partido(E, _, E2, _), leGano(E, E2)).
```

Responda a las siguientes consignas:

- Dónde aparece el concepto de orden superior. Justificar qué utilidad tiene, en particular dentro del contexto de la solución.
- En el programa hay un predicado que no es inversible. Indique:
 - cuál es ese predicado,
 - una consulta que no da las respuestas esperadas debido a la no inversibilidad,
 - por qué falla la inversibilidad, qué pasa en la definición del predicado que hace que no sea inversible,
 - cómo modificaría el programa para hacer al predicado inversible.