PARADIGMAS DE PROGRAMACIÓN – EXAMEN FINAL – 18/07/2015

Nota: Para aprobar es necesario obtener 8 puntos de 14 y al menos la mitad de los puntos en cada paradigma. En todas las respuestas sé puntual, sin perder el foco de lo que se pregunta. Las respuestas muy generales son tan malas como las incompletas.

Ejercicio A (6 puntos)

Se quiere calcular el promedio de las temperaturas en base a información climática. Se cuenta con esta solución:

```
#Mes (vi: dias)
>> temperaturaPromedio
|tot cant| tot:= 0. cant := 0.
dias do: [ :d |
   cant:= cant + 1.
   d conNubes ifTrue: [tot:= tot + d
temperatura * d cantNubes].
   d conNiebla ifTrue: [tot := tot + d
temperatura - d cantNiebla].
   d esNormal ifTrue: [tot := tot + d
temperatura] ].
^tot / cant.
```

```
#Día (vi: tRef tMod tInf tipo cantNubes
cantNiebla)
>> temperatura
 "extraña aunque precisa fórmula "
 ^(tRef * 1.15 + tMod sqrt)/ tInf
>> conNubes
 ^tipo = 'con nubes'.
>> conNiebla
 ^tipo = 'con niebla'.
>> esNormal
 ^self conNubes not & self conNiebla not
```

1) Proponer una nueva solución, de modo que se mejore (por lo menos) en términos de polimorfismo, delegación y encapsulamiento, indicando en qué lugar se usó cada uno. *(2 puntos)*

2) ¿El concepto de herencia es útil para resolver lo pedido? ¿Por qué? *(1 punto)*

3) Compare ambas soluciones (la dada y la del punto 1) en términos de expresividad y declaratividad *(1 punto)*

4) Si se quisiera que un día pueda cambiar (por ejemplo pasar de normal a nublado): ¿La solución desarrollada en el punto 1 permite esto? Si es posible, mostrar con código cómo se cambia un día normal a uno nublado. De lo contrario codificar un cambio sobre dicha solución para que sea posible, sin perder las mejoras realizadas inicialmente. *(2 puntos)*

Ejercicio B (4 puntos)

Se tiene las siguientes funciones:

```
f numeros = ((>40).head.filter even) numeros
g temperaturas = head (filter elevada temperaturas ) > 38.5
h nombres = head (filter (\ n -> last n == 's') nombres) > "perez"
```

1) Reescribir las tres funciones, sin repetir lógica, definiendo una función auxiliar. *(2 puntos)*

2) ¿Qué concepto se destaca en la nueva solución? Justificar su importancia para lograr lo pedido. *(1 punto)*

3) Reemplazar la expresión lambda de “h” por composición y aplicación parcial. *(1 punto)*

Ejercicio C (4 puntos)

En una base de conocimientos Prolog encontramos lo siguiente:

```
enProblemas(Persona) :-
  trabajaPara(Jefe, Persona),
  esMafioso(Jefe),
  encargo(Jefe,Persona,cuidar(Cuidado)),
  pareja(Jefe, Cuidado).
```

```
enProblemas(Persona) :-
   encargo(_,Persona,buscar(Buscado,_)),
   personaje(Buscado, boxeador).
enProblemas(butch).
```

1. Explicar qué restricciones mínimas deberían cumplir los predicados usados en enProblemas/1 para que pueda consultarse por quiénes cumplen el predicado enProblemas/1. (*1 punto*)

2. ¿Se está aprovechando el polimorfismo entre los distintos tipos de encargos (cuidar, buscar, etc)? (*1 punto*)

3. Se tiene un predicado para saber si una persona sólo hace encargos de cuidar. ¿Es posible resolverlo sin listas? En caso afirmativo codificar la nueva solución, si no indicar por qué no es posible. *(2 puntos)*

```
cuidador(Personaje):-
      encargo(_, Personaje, _),
      findall(Cuidado, encargo(_, Personaje, cuidar(Cuidado)), Cuidados),
      findall(Encargo, encargo(_, Personaje, Encargo), Encargos),
      length(Cuidados, Cant), length(Encargos, Cant).
```