Punto 1

Una empresa lleva el registro de sus empleados, para conocer los empleados ejemplares se tiene el siguiente código:

Smalltalk

```
>>Empresa (VI: empleados)
empleadosEjemplares
    ^empleados select: [ :empleado | empleado esEjemplar ]

>>Empleado (VI: tipo)
esEjemplar
     tipo = 'Jefe'     ifTrue: [ ^self esJefeEjemplar ]
     tipo = 'Gerente'  ifTrue: [ ^self esGerenteEjemplar ]
     tipo = 'Comun'    ifTrue: [ ^self esComunEjemplar ]

esJefeEjemplar
     ^false

esGerenteEjemplar
     ^false

esComunEjemplar
     ^false

>>Jefe subclase de Empleado (VI: empleadosACargo sector)
esJefeEjemplar
    ^ empleadosACargo size > 20
…
```

(Análogamente con el resto de los métodos, que se redefinen en cada subclase de Empleado, que son: Jefe, Gerente y Comun)

a. ¿Cuál es el error conceptual de la solución anterior? Justifique con los conceptos que apliquen al contexto.

b. Modifique el código anterior según su criterio. Justifique el porqué de sus cambios e indique qué conceptos aparecen en su solución.

Uno de los desarrolladores pensó otra solución para conocer los empleados ejemplares:

Smalltalk

```
>>Empresa (VI: empleados, empleadosEjemplares)
generarEmpleadosEjemplares
    empleadosEjemplares :=
         empleados select: [ :empleado | empleado esEjemplar ]

empleadosEjemplares
    ^empleadosEjemplares
```

c. La nueva solución introduce un concepto que antes no aparecía, ¿cuál es ese concepto?

d. Justifique con un ejemplo concreto en qué caso podría representar una ventaja la nueva solución.

Punto 2
Se tiene la siguiente base de conocimientos:

Prolog

```
actua(ricardoDarin).                actua(vandoVillamil).
actua(diegoPeretti).                actua(soledadVillamil).
actuaBien(ricardoDarin).            actuaBien(soledadVillamil).
```

a) ¿Cómo representaría ud. a alguien que no es actor, de manera que la consulta
`? actua(santoBiasatti)` devuelva `No`? ¿Con qué concepto está relacionado? Justifique.

Ahora se pide como requerimiento conocer los malos actores.

Prolog

```
actuaMal(X):-not(actuaBien(X)).
```

b) ¿Cuál es el problema de esta solución? Justifique a través de un ejemplo.

c) Corrija la solución (codifíquela nuevamente), indicando qué concepto aparece y porqué soluciona el inconveniente de b).

d) Relacione la nueva solución con el concepto de tipo.

e) Programar en Haskell las funciones que permitan saber si una persona
- Actúa
- Actúa bien
- Actúa mal

f) ¿Qué diferencias encuentra entre Prolog y Haskell para representar la información? Justificar.

Queremos conocer el conjunto/la lista de actores buenos.

g) Resuelva el problema en cada lenguaje aplicando el concepto de orden superior.

h) ¿Qué pasaría en cada caso si no existiera dicho concepto? Justifique.