1) Una aplicación en Smalltalk liquida los sueldos que la empresa tiene para pagar a sus empleados enviando el siguiente mensaje y obteniendo como respuesta el importe total:

```
unaEmpresa totalSueldosAPagar.
```

También, en otra parte de la aplicación se quiere saber cuál es el importe total de sueldos de los empleados y se envía el mismo mensaje.

```
>>Empresa (VI: empleados)
totalSueldosAPagar
    ^empleados inject: 0 into:
          [ :acum :empleado | acum
+ empleado pagarSueldo ]

>>Empleado (VI: cuenta, sueldo)
pagarSueldo
    cuenta depositar: sueldo.
    ^ sueldo.

>>Cuenta (VI: saldo)
depositar: unMonto
     saldo := saldo + unMonto
```

   a) ¿Qué problema presenta esta solución respecto al efecto colateral?
   b) Modifique la solución en base a lo contestado en a). Mostrar qué mensajes se deberían enviar en ambos casos.
   c) Se quiere resolver el mismo problema en Haskell. Codifique la solución correspondiente.
   d) ¿Qué inconvenientes tiene la solución c) para lograr la misma funcionalidad? Justifique e indique si de alguna manera puede salvar este inconveniente y lograr un resultado similar.
   e) ¿Qué herramienta del lenguaje Haskell se puede asociar el **inject: into:** utilizado en Smalltalk? ¿Qué diferencias y similitudes encuentra entre ambas herramientas?

2) Se necesita obtener el menor de dos números, de manera que el menor entre 4 y 7 es 4 y el menor de 8 y 9 es 8.

   a) Resuelva el problema en Smalltalk, Haskell y Prolog.
      Puede utilizar cualquier elemento del lenguaje, salvo la función min de Haskell y el mensaje min: de Smalltalk.
   b) Considerando que en Smalltalk existen diferentes tipos de números, representados por diferentes subclases de Number, y que en Haskell y Prolog hay también diferentes tipos de datos numéricos, analice las soluciones en términos de polimorfismo. Justificar la relación con este concepto.
   c) Analice la solución de Prolog en términos de inversibilidad.
      ¿Cuáles forma de consulta soporta y cuáles no? Justifique.
      ¿Es posible lograr que sea totalmente inversible?
      ¿Qué sucede en los otros lenguajes?
   d) Analice la solución de Haskell en términos de aplicación parcial. Considere tanto la definición que propuso como sus posibles usos. Justificar la relación con este concepto.
      ¿Qué sucede en los otros lenguajes?

3) Se quiere tener en Prolog una base de conocimientos de todos los empleados de una empresa y entre ellos poder saber quiénes tienen cargos gerenciales y quienes no.

   a) ¿Cómo modelaría dicha base? Piense al menos dos maneras diferentes. (Elegir una de ellas para continuar con los siguientes ítems.)
   b) Si quiere representar que el empleado Furattini no ocupa un cargo gerencial, ¿cómo lo resolvería? ¿con qué concepto está relacionado?
   c) Resolver el mismo problema en Haskell.
   d) ¿Qué diferencias y similitudes hay entre las dos soluciones? ¿Qué influencia tienen el paradigma y el lenguaje en esas soluciones?

4) Se quiere obtener el primer número capicúa a partir de un número dado. Las soluciones propuestas son:

En el paradigma funcional, en Haskell:

```
primerCapicua n =
   (head . filter capicua) [n, n + 1, ..]
```

En el paradigma imperativo, en pseudocódigo:

```
public int primerCapicua(int n) {
     boolean sigo = true;
     while (sigo) {
          if (capicua(n)) {
               sigo = false;
          } else {
               n++;
          }
     }
     return n;
}
```

Realizar un análisis comparativo relacionándolo con los siguientes conceptos:

a) Declaratividad
b) ¿Qué evita en cada solución que se hagan más cálculos de los necesarios?