Punto 1

Se presenta el siguiente requerimiento:

"Dado un conjunto de piratas, filtrar aquellos que son temibles. Decimos que un pirata es temible cuando logró alzarse con más de 10 botines".

a) Plantear un modelo de objetos que permita resolver el requerimiento, de manera de no plantear toda la solución como un script en un workspace, sino que pueda resolverse a través del envío de un mensaje con la colección como parámetro, y que dicho mensaje retorne la colección filtrada. Es decir:

```
coleccionFiltrada := objetoReceptor mensaje: coleccionDePersonas
```

Especifique su solución incluyendo un diagrama de clases, el código de esas clases y la forma de invocar el código desde un workspace.

b) Amplíe la resolución del punto a) de manera de soportar el siguiente requerimiento:
"Considerar para el filtro la siguiente funcionalidad:

- *de una colección de políticos, filtrar aquellos que son temibles (en su historial de cargos politicos ya tuvieron un cargo importante, ser presidente, diputado o senador es un cargo importante, es decirla importancia la determina el cargo)*
- *de una colección de goleadores, filtrar aquellos que son temibles (si tienen un promedio de gol de más de 0,5 por partido; se conoce la cantidad de goles y la cantidad de partidos jugados). "*

c) Indique cuáles son los conceptos más importantes del paradigma de objetos simplifican la implementación del punto b.
Justifique por qué se simplifica la implementación, indicando dónde aparecen esos conceptos en el código y qué ventaja proveen específicamente en este ejemplo.

Punto 2

En un teatro, se quiere saber en qué fila hay algún asiento libre, a partir de un número de fila que indica el espectador. Para ello se elaboraron dos soluciones que funcionan correctamente. Se asume la existencia de una función llamada hayLugar que recibe un número de fila y devuelve si es cierto que en esa fila hay algun lugar libre.

Paradigma funcional (Haskell):

```
filaConAsientoLibre n = head (filter hayLugar [n..])
```

Paradigma imperativo (seudo Pascal):

```
function filaConAsientoLibre (n: Integer): Integer
begin
     while not(hayLugar(n)) do
          begin
               n:= n + 1
          end
     filaConAsientoLibre:= n
end
```

Realizar un análisis comparativo a partir de los siguientes conceptos:

1. Declaratividad: ¿qué solución es más declarativa? Justificar.
2. ¿Cómo se evita en cada solución que se hagan más cálculos de los necesarios? No olvide mencionar los conceptos más importantes involucrados.
3. ¿Puede caer en un loop infinito? ¿En qué casos?

Punto 3

Dado el siguiente programa Prolog

```
padre(homero,bart).
padre(homero,lisa).
hermano(X,Y):- X \= Y, padre(Z,X), padre(Z,Y).
```

a. El predicado hermano/2, ¿Es inversible?. Justifique su respuesta referenciando al ejemplo. Exponga los ejemplos que muestran los casos más interesantes.

b. ¿Cuál va a ser la respuesta de Prolog si se carga el programa anterior y se hace esta consulta?

```
?- hermano(nico,mariano).
```

Justifique, mencionando el concepto más importante que lleva a obtener esa respuesta.

c. Agregamos ahora estas cláusulas al programa:

```
madre(marge,bart).
madre(marge,lisa).
```

Realice las modificaciones al programa anterior (codifique), para que el predicado hermano/2 relacione dos personas cuando tengan el mismo padre y la misma madre, y agregue el predicado hermanastro/2 que relacione los pares de personas que compartan al menos uno de los dos progenitores (ojo, si dos personas son hermanos, también serán considerados hermanastros).

Indique qué tienen de similar y qué tienen de distinto los predicados hermano/2 y hermanastro/2 así definidos, mostrando cómo se nota la diferencia en la implementación. Ambos predicados deben ser totalmente inversibles.