
Oooh, sí... "Bond... James Bond", el recién ascendido agente 007, necesita asegurar su victoria en la partida de Poker que se realizará en el Casino Royale, en la cual deberá enfrentar al "banquero de los terroristas": Le Chiffre. No puede permitirse una derrota en su primera misión como agente doble cero, y por lo tanto no se conforma con sus chances "naturales" en el Poker, así que nos pidió analizar el juego mediante un programa hecho en Haskell.

Para ayudar al agente 007 en su misión, tenemos el siguiente modelo:

Las cartas están definidas por tuplas **(número, palo)** siendo **número** un valor entre 1 y 13 y **palo** un valor de la siguiente lista:

```
palos = ["Corazones", "Picas", "Tréboles", "Diamantes"]
```

En el caso de los jugadores, tenemos los siguientes tipos definidos

```
type Carta = (Int, String)
data Jugador = Jugador String [Carta] String
```

Y las funciones constantes:

```
pokerDeAses = [(1,"Corazones"), (1,"Picas"), (1,"Tréboles"), (1,"Diamantes"), (10,"Diamantes")]
fullDeJokers = [(11,"Corazones"), (11,"Picas"), (11,"Tréboles"), (10,"Diamantes"), (10,"Picas")]
piernaDeNueves = [(9,"Corazones"), (9,"Picas"), (9,"Tréboles"), (10,"Diamantes"), (4,"Copas")]


jamesBond   = Jugador "Bond... James Bond" pokerDeAses "Martini... shaken, not stirred"
leChiffre   = Jugador "Le Chiffre" fullDeJokers "Gin"
felixLeiter = Jugador "Felix Leiter" piernaDeNueves "Whisky"
```

Una mesa de juego tiene a todos los jugadores participantes representados en una lista:

```
mesa1 = [jamesBond, leChiffre, felixLeiter]
```


Tenemos las siguientes funciones disponibles:

```
ocurrenciasDe x = length . filter (== x)
concatenar = foldl (++) []
```

Se pide definir las funciones indicadas, usando correctamente al menos una vez cada uno de los siguientes conceptos:

- Orden superior
- Lista por comprensión
- Aplicación parcial
- Composición

1.

a. **mayorSegun/3**, que dada una función y dos valores nos devuelve aquel valor que hace mayor a la función (en caso de igualdad, cualquiera de los dos).

b. **maximoSegun/2**, que dada una función y una lista de valores nos devuelve aquel valor de la lista que hace máximo a la función. **No usar recursividad.**

c. **sinRepetidos/1**, que dada una lista devuelve la misma sin elementos repetidos. Los elementos tienen que aparecer en el mismo orden que en la lista original (la primera ocurrencia de la lista de izquierda a derecha).

2.

a. **esoNoSeVale/1**, que se cumple para una carta inválida, ya sea por número o por palo (ver arriba cómo tiene que ser una carta).

b. **manoNegra/1**, que dado un jugador, nos indica si tiene una mano mal armada. Esto es cuando sus cartas no son exactamente 5, o alguna carta es inválida.

3. Dada una lista de cartas, hacer las funciones que verifican si las mismas forman un juego dado, según las siguientes definiciones:
   a. **par** --> tiene un número que se repite 2 veces
   b. **pierna** --> tiene un número que se repite 3 veces
   c. **color** --> todas sus cartas son del mismo palo
   d. **fullHouse** --> es, a la vez, par y pierna
   e. **poker** --> tiene un número que se repite 4 veces
   f. **otro** --> se cumple para cualquier conjunto de cartas

   Ejemplos:

```
?- par [(1,"Corazones"), (1,"Picas"), (3,"Tréboles"), (4,"Diamantes"), (10,"Diamantes")]
True
?- par [(1,"Corazones"), (1,"Picas"), (1,"Tréboles"), (1,"Diamantes"), (10,"Diamantes")]
False
```

   Como pueve verse, cuando se indica la cantidad de veces que tiene que repetirse, tiene que ser la cantidad *exacta*.

4. **alguienSeCarteo/1**, dada una lista de jugadores. Sabemos que alguien se carteó cuando hay alguna carta que se repite, ya sea en un mismo jugador o en distintos.

5.

a. Dada la siguiente lista de valores para los distintos juegos: **valores = [(par,1), (pierna,2), (color,3), (fullHouse,5), (poker,5), (otro, 0)]**
Definir **valor/1** que, dada una lista de cartas, nos indique el valor del mismo, que es el máximo valor entre los juegos que la lista de cartas cumple.

b. **bebidaWinner/1**, que dada una lista de jugadores nos devuelve la bebida de aquel jugador que tiene el juego de mayor valor, pero sin considerar a aquellos que tienen manos mal armadas.

6. Realizar las consultas que indiquen:
    a. El nombre del jugador que está tomando la bebida de nombre más largo.
    b. El jugador con mayor cantidad de cartas inválidas.
    c. El jugador de nombre más corto.
    d. El nombre del ganador de una mesa, que es aquel del jugador con el juego de mayor valor.
    **Nota:** No definir funciones auxiliares para este punto. Construir las consultas únicamente en base a las funciones definidas para puntos anteriores.

7.
    a. Implementar una función de ordenamiento **`ordenar/2`**, que dado un criterio y una lista me devuelva la misma lista con sus elementos ordenados en base al criterio.

```
> ordenar (>) [5, 7, 2, 4, 6]
[2, 4, 5, 6, 7]
```

    **Tip**: Notar la diferencia con el punto 1, ya que la función-parámetro devuelve un booleano en lugar de uno de los valores comparados.

    b. Implementar las funciones para los juegos "escalera" y "escalera de color".
        i. Una **`escalera`** es una sucesión de números con las 5 cartas. ¡Cuidado! Las cartas pueden estar desordenadas. Por ejemplo, si tengo las cartas 6, 3, 4, 2, 5 (no importa el palo) eso es una escalera porque tiene las cartas del 2 al 6.
        ii. Una **`escaleraDeColor`** es como una escalera pero que tiene todas las cartas del mismo palo.