A un grupo de timberos pdepianos le dio ganas de jugar a las Bazas[1] (mejor conocido como La Podrida), en donde lo importante no es obtener más o menos bazas que el resto, sino acertar de antemano cuántas se ganarán durante esa mano. Cuando un jugador logra llevarse la cantidad exacta de bazas pedidas para esa mano, gana puntos extra, además de lo que le corresponda por su cantidad de bazas. Al finalizar el juego (la cantidad de manos es fija), gana el jugador que más puntos haya acumulado.

Decidimos armar un programa en Haskell para modelar una partida de Bazas, teniendo la información de las manos transcurridas y de la mano actual.

Las manos transcurridas tienen el formato

```
[(cantidadCartas,[(jugador, bazasPedidas, bazasGanadas)])]
```

```
manosTranscurridas = [
      (1,[("Maiu",0,1),("Guille",1,0),("Nico",0,0),("Adriel",1,0)]),
      (3,[("Maiu",1,1),("Guille",1,2),("Nico",1,0),("Adriel",1,0)]),
      (5,[("Maiu",2,1),("Guille",1,1),("Nico",2,2),("Adriel",1,1)]),
      (7,[("Maiu",3,5),("Guille",1,0),("Nico",0,0),("Adriel",1,0)]),
      (9,[("Maiu",4,4),("Guille",2,1),("Nico",2,3),("Adriel",1,1)]),
      (11,[("Maiu",5,5),("Guille",4,5),("Nico",1,1),("Adriel",0,0)]),
      (10,[("Maiu",3,1),("Guille",2,3),("Nico",7,6),("Adriel",2,0)]),
      (8,[("Maiu",2,2),("Guille",1,1),("Nico",2,3),("Adriel",1,2)]),
      (6,[("Maiu",0,1),("Guille",1,3),("Nico",2,2),("Adriel",1,0)])]
```

La mano actual se modela como una tupla de la forma:

```
      (triunfo[2],(cantidadCartas, [(jugador, bazasPedidas,
cartasEnMano)]))
```

```
manoActual = ("oro", (4,
      [("Maiu",1,[(8,"basto"),(1,"oro"),(2,"oro"),(2,"copa")]),
      ("Guille",2,[(1,"basto"),(4,"oro"),(5,"copa"),(12,"espada")]),
      ("Nico",0, [(2,"espada"),(3,"copa"),(2,"copa"),(7,"basto")]),
      ("Adriel",1,[(10,"basto"),(9,"oro"),(7,"oro"),(6,"espada")])]))
```

Se desea resolver los siguientes problemas sobre el contexto planteado, demostrando el conocimiento de los conceptos del paradigma funcional:

- Composición
- Orden superior
- Aplicación Parcial
- Listas por comprensión

Solo vale resolver mediante recursividad **UNO** de los ejercicio dados. Sin embargo, es posible resolverlos todos sin ella.

Nota: NO OLVIDEN LEER LAS NOTAS AL PIE DE PÁGINA

---

1 Una **baza** es el montón de cartas que recoge un jugador en ciertos juegos de naipes (como el Bridge y el Corazones), generalmente después de jugar la carta más alta.

2 El triunfo es el palo más valioso para una determinada mano. Un 2 de triunfo le gana a un ancho de cualquier otro palo, más allá de ser numéricamente inferior.

1a. Dada una vuelta, determinar cuántos puntos se otorgan sabiendo que cada baza ganada vale un punto y, en caso de haber cumplido con las basas pedidas, se gana un plus de 10 puntos más.

```
> puntosGanados ("Maiu",1,1)
      11 (1 por la baza y 10 por haber acertado)
```

1b. Saber si una carta le gana a otra dado un triunfo considerando que:

- El as es la carta más alta y el 2 la más baja
- Si una carta es de triunfo y la otra no, gana seguro el triunfo, de lo contrario vale la regla anterior
- Hay que seguir el palo. Si el que es mano tira una carta de cualquier palo y el otro tira una de otro palo diferente que tampoco es triunfo, el segundo pierde automáticamente (se considera "mano" al primer parámetro).

```
> leGana (8, "basto") (7, "copa") "copa"
      False (porque el segundo tiró triunfo)

> leGana (5, "espada") (12, "oro") "basto"
      True (porque el segundo no siguió el palo ni tiró un triunfo)
```

2a. Obtener todos los resultados de un jugador a partir de una lista de manos. (Miren bien, esta función les va a servir dos veces más en este parcial, no repitan el código!)

```
> resultadosJugador "Maiu" manosTranscurridas
      [("Maiu",0,1),("Maiu",1,1),("Maiu",2,1),.... etc]
```

2b. Calcular los puntos de un jugador a partir de una lista de manos.

```
> puntos "Maiu" manosTranscurridas
      61 (1+11+1+5+14+15+1+12+1)
```

3. Saber si una determinada mano está mal pedida. Esto pasa cuando la sumatoria de bazas pedidas entre todos los jugadores es igual a la cantidad de cartas de esa mano, porque sino puede darse que todos cumplan a la vez y eso le quita la gracia al juego.

```
> manoMalPedida (head manosTranscurridas)
      False

> manoMalPedida (manosTranscurridas !! 4)
      True (4+2+2+1=9)
```

4a. Obtener las cartas de un jugador en la mano actual:

```
> cartas "Nico" manoActual
      [(2,"espada"),(3,"copa"),(2,"copa"),(7,"basto")]
```

4b. Saber si un jugador tiene una carta del triunfo en la mano actual (ojo, no repetir código)

```
> tieneTriunfo "Nico" manoActual
      False
```

5a. Saber cuántas veces cumplió con sus bazas pedidas un jugador, dada una lista de manos (de nuevo, es más fácil si usás lo que ya hiciste!)

```
> vecesCumplio "Maiu" manosTranscurridas
      4
```

5b. Realizar la función **`mejorSegunCriterio`** que reciba un criterio, una lista de manos y una lista de

jugadores; y devuelva el mejor jugador según ese criterio.
Naturalmente utilizaremos una función para representar el criterio, en este caso esa función deberá recibir por parámetro a un jugador y la lista de manos y deberá devolver un número, conceptualmente sería: "Jugador -> [Mano] -> Número".

Para eso nos vendría bien tener una lista de jugadores, por ejemplo:

```
jugadores = ["Maiu","Guille","Nico","Adriel"]
```

Y luego la función podría ser utilizada así:

```
> mejorSegunCriterio vecesCumplio manosTranscurridas jugadores
    "Nico"
```

5c. Utilizar la función **`mejorSegunCriterio`** en conjunto con la función **`puntos`** (2b) para saber cuál de los jugadores va ganando el partido.