Night Club

Se cuenta con la siguiente lista de ingresantes a una discoteca:

```
discotequers = [("fer", 500, 10, [("Coca cola", 1), ("Sprite Zero", 1)]),
          ("mati", 1670, 2000, [("Cerveza", 2)]),
          ("german", 40000, 200000, [("Grog XD", 25), ("Cerveza", 1)]),
          ("flor", 5000, 15, [("Grapa", 1)])]
```

Cada tupla representa:

- El nombre de la persona que entró a la discoteca
- La cantidad de levante que tiene
- El nivel de aguante de alcohol
- La lista de tragos que tomó, cada trago se representa como una tupla que define
  - El nombre de la bebida
  - La cantidad de vasos que tomó

Por otra parte se tiene la lista de tragos:

```
bebidas = [("Coca cola", 0), ("Grog XD", 350), ("Sprite Zero", 0),
    ("Cerveza", 10), ("Grapa", 40)]
```

El formato que sigue la tupla es:

- Nombre de la bebida
- Graduación alcohólica de cada vaso

Se cuenta con estas funciones:

```
nombre (n, _, _, _) = n
levante (_, l, _, _) = l
aguante (_, _, a, _) = a
tragos (_, _, _, t) = t
bebida = fst
graduacion = snd
find criterio = head . filter criterio
```

Codificar las funciones indicadas a continuación. En la resolución tenga presente que deben aparecer aplicados, al menos una vez, los siguientes conceptos:

- Aplicación parcial
- Composición
- Funciones de orden superior
- Listas por comprensión

No se puede usar recursividad, a menos que esté indicado en el punto.

Se pide

1)

a. Encontrar los datos de una persona que ingresó a la discoteca en base al nombre

```
>datosDe "german"
("german",40000,200000,[("Grog XD",25),("Cerveza",1)])
```

b. Resolver la graduación alcohólica de un trago

```
>graduacionAlcoholica "Grog XD"
350
```

c. Resolver la cantidad de alcohol en sangre que tiene una persona

```
>alcoholEnSangreDe "german"
8760 (350 del Grog XD * 25 + 10 de la Cerveza * 1)
```

2)

a. Saber si una persona está borracha (si el aguante es < que el alcohol en sangre)

```
>estaBorracho "german"
False
```

b. Determinar el nivel de levante real de una persona, esto se determina así: si la persona está borracha, es su nivel de levante base – el nivel de alcohol en sangre, si no está borracha es su nivel de levante base + el nivel de alcohol en sangre. **Nota:** evitar repetición de código.

```
> nivelLevanteReal "german"
48760 (40000 + 8760 porque no está borracho)
> nivelLevanteReal "flor"
4960  (5000 – 40 porque está borracha)
```

3)

a. Determinar entre dos personas quién es mejor, en base a un criterio pasado como parámetro (el criterio aplica sobre una tupla persona):

```
>quienEsMejor aguante "flor" "german"
"german"
```
(porque tiene más nivel de aguante, 200.000 > 15 de flor)

b. Usar la función anterior en una consulta para encontrar, dadas dos personas cualesquiera

i. El que tomó más variedad de bebidas (ej: Fernando tomó 2 bebidas, más que Mati que tomó sólo cerveza)

ii. El que tomó más vasos (ej: Mati tomó 2 vasos, Flor tomó sólo 1, no importa la bebida, hay que sumar los vasos)

Nota: no se pueden definir expresiones lambda ni funciones auxiliares (el criterio a pasar debe obtenerse solamente componiendo funciones existentes)

4) Saber si la lista de integrantes está ordenada de menor a mayor en base a un criterio que pasamos por parámetro.
Nota: Únicamente en este punto se puede usar recursividad

```
>estaOrdenada aguante discotequers
False (10 <= 2000 <= 200.000 pero Flor tiene 15)
```

5)

a. Saber si un integrante está "roto", esto es cuando mezcló 2 ó más bebidas alcohólicas.

```
>estaRoto "german"
True
```
(mezcló Grog XD y Cerveza, para el resto da falso)

b. Definir estoEsUnDescontrol, que se cumple cuando todas las personas dadas están rotas, según lo definido anteriormente.

```
>estoEsUnDescontrol discotequers
False
```
(no todos están rotos)

6) Inferir los tipos de la función funcionHeavy

```
funcionHeavy w x y z | elem w x  = filter y x
                     | otherwise = map z x
```