Paradigmas de Programacíón Recuperatorio Funcional - 18/12/2010
Nombre: Legajo: Profesor:

Administración de maldad

Una persona importante (cuya identidad no es necesaria revelar), nos pidió la construcción de un software de gestión para la distribución y redireccionamiento de las regalías por las fiestas.
Sin importar mucho el anterior preámbulo sin sentido, se nos pide representar personas y acciones en Haskell.

Una acción está representada por una función que recibe una persona y devuelve una persona.

```
teRobaron = perderDinero
abrazo = variarFelicidad 42
variarFelicidad cantidad (nombre, felicidad, maldad, dinero) =
      (nombre, felicidad+cantidad, maldad, dinero)
variarDinero platita (nombre, felicidad, maldad, dinero) =
      (nombre, felicidad , maldad, max (dinero+platita) 0)
variarMaldad cant (nombre, felicidad, maldad, dinero) =
      (nombre, felicidad, maldad + cant, dinero)
perderDinero x = variarDinero (x * -1)
```

Una persona está representada por una tupla de 4 elementos (nombre,felicidad,maldad,dinero)

Ejemplo

```
> abrazo ("sergei",34,4,1345)
("sergei",76,4,1345)
```

```
todasLasPersonas =
      [("sergei",34,4,1345),("stvan",5,110,666666),("tavo",0,73,20),...]
```

```
find f xs = head (filter f xs)
fst4 (x,_,_,_) = x
trd4 (_,_,z,_) = z
```

```
> div 5 3
1
```

Nota: Donde vean... (3 puntitos) significa que eso es lo que tienen que completar

1. Crear la acción **trabajó** que recibe una persona, devuelve la persona con las modificaciones y hace lo siguiente.
Si la felicidad de la persona es < 30 le aumenta en 5 la maldad, sino le suma 3 a su felicidad.
2. Crear la acción **cobrarSueldo/2**, que recibe una cantidad de dinero y una persona. Consiste en aumentar la felicidad en 2 y aumentar el dinero con ese sueldo

```
> cobrarSueldo 3500 ("sergei",34,4,1345)
("sergei",36,4,4845)
```

a. Resolverlo utilizando pattern matching y sin funciones auxiliares (suyas o que ya vienen)
b. Resolverlo con composición (TIP: Ver las funciones que vienen definidas arriba de todo)

3. Representar el año de Sergei como una lista de acciones.
Sus 4 acciones del año fueron:

- Recibir un abrazo
- Cobrar $2500
- Que le robaran $200
- Insultó (esto aumenta la maldad en 5 puntos)
- Trabajó

```
añoDeSergei = ...
```

Un año es una lista de acciones. El tipo de la lista debe ser:

```
añoDeSergei = [(String,Int,Int,Int) -> (String,Int,Int,Int)]
```

4. Definir cuál es el **tipo** de la siguiente función

```
maximoSegun _ [ e ] = e
maximoSegun f (x:xs)
      | f x > f maximoDeXs = x
      | otherwise = maximoDeXs
      where maximoDeXs = maximoSegun f xs

maximoSegun :: ...
```

Pueden usar maximoSegun en cualquier punto que crean necesario

5. Saber cómo quedaría una persona después de que pasó su año

```
> comoQuedarias añoDeSergei ("sergei",34,4,1345)
("sergei",78,9,3645)
```

6. **elMasMalo/2** Dada una lista con nombres de personas y un año (como se definió en el punto 3), saber cuál es la persona que quedaría con el mayor nivel de maldad. Esta función devuelve a la persona (la tupla, no el nombre) que cumpla con lo anterior pero devuelve a la persona en su estado original (sin aplicarle todas las acciones del año).

```
> elMasMalo ["sergei","stvan","tavo"] añoDeSergei
("stvan",5,110,666666)
```

7. Dada una lista de nombres de personas y un año, saber cuál es la persona que más *algo*.

```
elQueMasAlgo nombres año f = ...
```

Tienen que hacer una sola función que resuelva esto y explicar qué función pasarían como parámetro para resolver:
a. El que quedaría con el mayor nivel de maldad (si, lo mismo que el punto 6 pero usando la función del punto 7)
b. El que quedaría con el menor nivel de maldad
c. El que quedaría con el mejor promedio entero (usando div) entre felicidad y maldad

BONUS 7c. El que quedaría con la menor variación de maldad
(la diferencia entre la mayor maldad que tuvo y la menor maldad que tuvo)