PARADIGMAS DE PROGRAMACIÓN – CURSO K2054 – 2do CUATRIMESTRE 2006
NOMBRE Y APELLIDO: **HOJA 1 /**

En la agencia de Modelos "Pancho & Coca S.A."[1] se tiene la siguiente información sobre las modelos:

modelos = [("nicolle", 88, 59, 91), ("sofia", 87, 59, 91), ("nazarena", 102, 62, 103),
("ximena", 90, 63, 90), ("wanda", 88, 58, 92), ("victoria", 91, 62, 92)]

Se pide:

a) Definir una función **`diferencia`** que devuelva la distancia de una modelo respecto al ideal del cuerpo femenino (90-60-90).

diferencia ("nicolle", 88, 59, 91)
4 (le faltan 2 para llegar a los 90 de busto, 1 para llegar a los 60 de cintura y le sobra 1 de cola)

Nota: considerar diferencia tanto por faltante como por sobrante (no deben compensarse las diferencias de más y de menos, sino que todas suman)

b) Definir la función **`filtrarGordas`** que filtre las modelos que tienen más de 60 cms. de cintura.

filtrarGordas modelos
[("nazarena", 102, 62, 103), ("ximena", 90, 63, 90), ("victoria", 91, 62, 92)]

Resolverlo aplicando filter + expresiones lambda o definición local de una función.
Justifique dónde aparece el concepto función de orden superior y qué ventaja tiene.

c) Definir la función **`puedenSerVedettes`** que devuelve los nombres de las modelos que tienen más de 90 cms de busto y más de 90 cms de cola.

puedenSerVedettes modelos
["nazarena", "victoria"]

Resolverlo utilizando:

- Una definición recursiva
- Listas por comprensión

Compare ambas soluciones justificando las principales diferencias.

d) Definir la funcion **`desvioGeneral`**, que nos devuelva la suma de las diferencias de una lista de modelos respecto al ideal del cuerpo femenino (90-60-90).

desvioGeneral modelos
50 (4 de nicolle, 5 de sofia, 27 de nazarena, 3 de ximena, 6 de wanda y 5 de victoria)

Resolverlo utilizando:

- Map o Foldl
- Funciones recursivas

e) Explique en alguna función anteriormente definida dónde aplicó composición y por qué.

f) Definir el tipo de la siguiente función:

```
funcionLoca a b c | a c > b c = a
                  | otherwise = b
```

Justificar dicha decisión.

Tips:

- al comparar no vale definir los dos conceptos de memoria, sino que hay que exponerlos dejando en claro similitudes y diferencias.
- Definir las funciones *como pide el enunciado*.
- Numerar y firmar todas las hojas.

---

[1] También conocida como "Ibáñez & Sarli Inc."

PARADIGMAS DE PROGRAMACIÓN – CURSO K2054 – 2do CUATRIMESTRE 2006
NOMBRE Y APELLIDO: **HOJA 1 /**

Respuesta	¿Qué se evalúa?
a)	Uso de tuplas, definición de funciones básicas y auxiliares
b)	Funciones de orden superior, Expresiones lambda/definición local vs. funciones auxiliares, declaratividad
c)	Listas por comprensión, funciones recursivas, declaratividad
d)	Aplicación de funciones de orden superior, declaratividad
e)	Composición
f)	Tipos, Función como valor de primer orden, Definición de una función de orden superior

Soluciones

a) diferencia (_, busto, cintura, cola) = abs (busto - 90) + abs (cintura - 60) + abs (cola - 90)

b) filtrarGordas xs = filter (\(_, _, cintura, _) -> cintura > 60) xs
la otra
filtrarGordas2 xs = filter estaGorda xs where estaGorda (_, _, cintura, _) = cintura > 60

Función de orden superior aparece en filter, cuando el criterio para determinar si una modelo es gorda lo da la función que justamente hay que definir. Es un bonus muy apreciado hablar sobre expresiones lambda para que no tenga que definir una función que sólo utilizaría una vez.

c)
puedenSerVedettes listaModelos = [modelo | (modelo, busto, _, cola) <- listaModelos, busto > 90, cola > 90]
Vale también usar funciones auxiliares si no se animan a hacer pattern matching.
La opción recursiva es:

```
puedenSerVedettes2 [] = []
puedenSerVedettes2 ((modelo, busto, _, cola):xs)
        | busto > 90 && cola > 90  = modelo:puedenSerVedettes2 xs
        | otherwise                = puedenSerVedettes2 xs
```

En la respuesta esperamos que hablen de declaratividad, que no hay algoritmo con listas por comprensión y que la solución recursiva no es tan clara (hay que escribir un poco más) que en la otra solución, aunque no siempre puedo garantizar eso. Lo importante está en la secuencia de pasos y en cuánto me acerco al lenguaje que maneja el matemático (con listas por comprensión me acerco más). Ambas soluciones son declarativas respecto al pseudocódigo.

d) desvioGeneral xs = sum (map diferencia xs)
También vale: desvioGeneral = sum . map diferencia

desvioGeneral2 xs = foldl (\x y -> x + diferencia y) 0 xs
en lugar de expresión lambda se puede usar función auxiliar o definición local.

Utilizando una definición recursiva:
desvioGeneral3 [] = 0
desvioGeneral3 (x:xs) = diferencia x + desvioGeneral3 xs

e) En el punto a) utilizamos abs de lo que nos devuelve busto - 90. A su vez el resultado de abs se utiliza con el operador + para obtener la diferencia.
En el punto d) utilizamos sum de lo que nos devuelve map diferencia xs.
Estaría bueno que muestren el gráfico de las dos cajas.

f) funcionLoca :: (a->b) -> (a->b) -> a -> (a ->b)
Lo interesante es que se aviven que devuelve una función y que justifiquen por qué usan tipos genéricos.

Opcional (no exigido)
Vale decir que como los voy a comparar con el operador (>), los b son **Ord**enables. O sea, también puedo decir: funcionLoca :: Ord b => (a->b) -> (a->b) -> a -> (a ->b)