UNIVERSIDAD NACIONAL SAN AGUSTIN DE AREQUIPA

EXAMEN DE LABORATORIO DE MATEMÁTICAS DISCRETAS 2

2006 – PAR

INGENIERÍA DE SISTEMAS Lic. WILBER RAMOS LOVÓN

Los ejercicios de esta práctica piden la definición de funciones al estilo de los programas funcionales de Haskell. En general, no hay una única manera de definir una función. Sin embargo, nos interesa que las soluciones elegidas sean claras, comprensibles, compactas (breves, de ser posible), completas (que contemplen todos los casos de interés) y declarativas (que estén más cerca del qué y no del cómo).
Cuando resulte conveniente, se pueden definir funciones auxiliares para simplificar la definición de la función principal. También se pueden usar funciones definidas previamente en otros ejercicios. Dar explícitamente el tipo de una función (el tipo de su domino y su imagen) permite detectar errores tempranamente. No hay que olvidar que el valor que devuelve una función depende sólo de sus argumentos.

---

1. Teniendo la función de orden superior:

```
itera :: Int->(a->a)->a->a
itera 0 f x = x
itera n f x = f ( itera  (n-1) f  x)
```

   Defina la función potencia usando itera, la función potencia que recibe dos enteros y retorna el primer parámetro elevado al segundo parámetro.

2. Definir las siguientes funciones sobre listas de caracteres:

   a. sacarTodos :: [Char] -> [Char] -> [Char], que elimina todas las apariciones de la primer lista en la segunda. Por ejemplo sacarTodos [c, a] [a, c, a, d, c, a] es [a, d].
   b. sacarBlancosRepetidos :: [Char] -> [Char], que reemplaza cada subsecuencia de blancos contiguos del primer parámetro por un solo blanco en el segundo parámetro.
   c. contarPalabras :: [Char] ->Integer, que devuelve la cantidad de palabras del parámetro.
   d. palabraMasLarga :: [Char] -> [Char], que devuelve la palabra más larga del parámetro.
   e. aplanar :: [[Char]] -> [Char], que a partir de una lista de palabras arma una lista de caracteres concatenándolas.
   f. aplanarConBlancos :: [[Char]] -> [Char], que a partir de una lista de palabras, arma una lista de caracteres concatenándolas e insertando un blanco entre cada par.
   g. aplanarConNBlancos :: [[Char]] -> Integer -> [Char], que a partir de una lista de palabras, arma una lista de caracteres concatenándolas e insertando n blancos entre cada par (n debe ser no negativo).

   ¿Estas funciones son polimórficas o sobrecargadas?

3. Dada una lista L, definir una función que devuelva:

   a. la lista L ordenada ascendentemente
   b. la lista L ordenada descendentemente
   c. una lista de listas tal que cada lista tiene todos sus elementos iguales y longitud igual a la cantidad de apariciones de ese elemento en L, además la lista resultante esta "ordenada" ascendentemente. Por ejemplo, L = [8, 5, 5, 4, 9, 4, 4, 4, 8] la función debería devolver [[4,4,4,4],[5,5],[8,8],[9]].

4. Implementar el nuevo tipo Vector y las funciones:

   a. abscisa :: Vectores -> Float
   b. ordenada :: Vectores -> Float
   c. igualX (Vector v1 v2) con resultado un Bool.
   d. igualY (Vector v1 v2) con resultado un Bool.
   e. colineales (: Vector v1 v2) con un resultado un Bool.
   f. norma (Vector v)
   g. productoEscalar (Vector v1, v2)

5. Implementar el nuevo tipo MatricesCuadradas y las funciones :

   a. dimensión :: MatrizCuadradas -> Int ; devuelve el tamaño de la matriz
   b. valor :: MatrizCuadrada -> Fila ->Columna -> Float, que devuelve el valor de la casilla indicada. Esta función está indefinida cuando Fila y Columna están fuera del rango válido. Los tipos Fila y Columna son sinónimos de Int.
   c. todosIguales :: MatrizCuadrada -> Bool, que indica si todos los elementos de la matriz cuadrada tienen el mismo valor.
   d. identidad :: MatrizCuadrada -> Bool, que indica si la matriz es una matriz identidad.
   e. máximo :: MatrizCuadrada -> Int, que devuelve el valor máximo de la matriz.
   f. mínimo :: MatrizCuadrada -> Int, que devuelve el valor mínimo de la matriz.

6. Implementar el nuevo tipo Polinomio y las funciones :

   a. grado :: Polinomio -> Float
   b. coeficiente :: Polinomio -> Int -> Float
   c. evaluar :: Polinomio -> Float -> Float , que devuelve el valor del polinomio en el punto dado.

7. La salida de un escaner de un supermercado obtenida a partir de una bolsa de la compra produce una lista de codigos de barras, como por ejemplo: [1234,4719,3814,1112,1113,1234] que se quiere convertir en una factura del tipo:

```
Supermercados Haskell
    Aceite oliva, 1 l..... .        ..5.40
    Barritas de merluza..  ..  ..11.21
```

Mermelada naranja..... .. .0.56
Bal_on playa (Gigante).....1.33
Item desconocido..... ...0.00
Aceite oliva, 1 liltro 5.40
Total..... .....23.90

Utilice :

type TipoSupermercado = [(CodBarras,Nombre,Precio)]

La base de datos ejemplo:

```
supermercado :: TipoSupermercado
supermercado = [ (4719, "Barritas de merluza", 11),
(5643, "Panales, talla 2", 20),
(3814, "Mermelada naranja", 56.50),
(1111, "Balón playa (Mediano)", 21),
(1112, "Balón playa (Gigante)", 133),
(1234, "Aceite oliva, 1 litro.", 23.50) ]
```

Para hacer un programa que convierta una lista de códigos de barras en una lista de pares (Nombre, Precio). A continuación, esta lista debe convertirse en un string que se pueda imprimir como se muestra al principio. Para ello, hacer dos definiciones nuevas:

```
type ListaCodigos = [CodBarras]
type TipoFactura = [(Nombre, Precio)]
```

Luego, defina las funciones:

```
hacerFactura :: ListaCodigos -> TipoFactura
que toma una lista de códigos de barras y obtiene una lista de pares nombre/precio,
formatearFactura :: TipoFactura -> String
que toma una lista de pares nombre/precio y obtiene una factura formateada, y
producirFactura :: ListaCodigos -> String
que combina los efectos de hacerFactura y formatearFactura, de modo que
producirFactura = formatearFactura . hacerFactura
```

8. En un proyecto de desarrollo de un sistema de gestión para videoclubs, se te pide, como parte del staff de programadores, que definas ciertas funciones del sistema. Se te pide definir los tipos : VideoClub, Cliente y Película ; y las siguientes funciones:

   a. clientes :: VideoClub -> [Cliente], que devuelve una lista con los clientes registrados en el videoclub.
   b. películas :: VideoClub -> [Película], que devuelve una lista con las películas que están en el catálogo de videoclub, independientemente de si está disponible o no.
   c. cantidadDeCopias :: VideoClub -> Película -> Int, que devuelve cuántas copias de la película tiene el videoclub (esta función se indefine, o sea da error, si la película no pertenece al catálogo del videoclub).
   d. peIıcuIasAIquiIadas :: Videoclub -> Cliente -> [Película], que devuelve una lista de películas que el cliente tiene alquiladas en el videoclub (esta función se indefine si el cliente no está registrado en el videoclub). El cliente puede alquilar más de una copia de la misma película.
   e. alquiló :: VideoClub -> Cliente -> Película -> Bool, que indica si el cliente alquiló la película.

f. copiasAlquiladas :: VideoClub -> Película -> Int, que devuelve la cantidad de copias de la película dada que están alquiladas en el videoclub.
g. disponibleParaAlquilar :: Videoclub -> Película -> Bool, que indica si una película está disponible para ser alquilada en el videoclub (por ejemplo, una película puede estar en un videoclub pero no estar disponible porque todas sus copias están alquiladas, o no estar disponible porque no está en el catálogo).

Dada la crisis económica los videoclubs más pequeños han decidido unirse en una red para brindar servicios adicionales al cliente como, por ejemplo, distribución a domicilio y compartir el conjunto de películas que disponen. Debido a esto se pide que definas las funciones :

h. disponibleEnLaRed :: [Videoclub] ->Película -> Bool, que dice si la película está disponible en al menos alguno de los videoclubs de la red.
i. másAlquiladaEnLaRed :: [Videoclub] -> Película, que devuelve la película más alquilada en todos los videoclubs de la red.
j. clientesRaros :: [Videoclub] ->[Cliente], que devuelve la lista de clientes que tienen alquilada la misma película en distintos videoclubs.

9. Decidir si las siguientes funciones son de orden superior o no, y en qué casos están currificadas. Ejemplificar.
   a. fUno :: ((Int,Int)->Int)->Int->Int->Int
      fUno f x y = f (x,y)
   b. fDos :: (Char->Char->Bool)->(Char->Char)->(Int->Char)->Char->Int->Bool
      fDos f1 f2 f3 a b = f1 (f2 a) (f3 b)
   c. fTres :: (Char,Char,Char) ->Bool
      fTres (c1,c2,c3) = (c1==c2) && (c2==c3)