CI4251 - Programación Funcional Avanzada
Tarea 1

Ernesto Hernández-Novich
86-17791
<emhn@usb.ve>

Mayo 2, 2013

1. Fold abstracto

- **(1 punto)** – considere la función `dropWhile` provista en el Preludio y en `Data.List`. Provea una implantación de `dropWhile` empleando el `fold` más apropiado según el caso.
- **(2 puntos)** – Provea una implantación de `foldl` usando `foldr`.

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl = {- Algo con foldr -}
```

  Incluya un diagrama que ilustre el funcionamiento de su implantación.

2. Foldable y Functor

Considere el tipo de datos

```
data Dream b a = Dream a
               | Limbo (b,a)
               | Within a (Seq (Dream b a))
               | Nightmare b
               deriving (Show)
```

- **(1 puntos)** – Construya la instancia `Functor` para el tipo `Dream b`.
- **(2 puntos)** – Construya la instancia `Foldable` para el tipo `Dream b`.

3. Monoid

Considere el tipo de datos `(Data.Map k v)` comentado en clase, que tiene algún tipo de datos `k` como clave y sobre el cual queremos permitir *múltiples valores* asociados a una clave.

Proponga un tipo de datos concreto apoyado en `Data.Map` que permita esa funcionalidad, y entonces:

- **(2 puntos)** – Construya la instancia `Monoid` para este tipo de datos. En la instancia queremos que al combinar dos `Map`, si hay claves repetidas, se *unan* los valores asociados.
- **(1 punto)** – Escriba un ejemplo de uso con al menos *tres* tablas involucradas, que contengan claves *repetidas* cuyos valores deban combinarse para ejercitar el `Monoid` a la medida.

4. Zippers

Considere el tipo de datos

```
data Tree a = Leaf a | Node a (Tree a) (Tree a) (Tree a)
```

(3 puntos) – diseñe un zipper seguro para el tipo `Tree` proveyendo todas las funciones de soporte que permitan trasladar el foco dentro de la estructura de datos, así como la modificación de cualquier posición dentro de la estructura.

```
data Breadcrumbs a = undefined

type Zipper a = (Tree a, Breadcrumbs a)

goLeft   ::
goRight  ::
goCenter ::
goBack   ::
tothetop ::
modify   ::
focus    ::
defocus  ::
```