LP2 - Linguagem Funcional
Primeiro Exame Escrito

Sérgio Soares
02 de Março de 2002

1. (2.0) indique, explicando, se as expressões abaixo estão corretas. Em caso afirmativo indique seu tipo e, se for adequado, seu resultado.
   (a) (0.5) `map (fold (+)) [[1,2],[3,4], [5]]`
   (b) (0.5) `fold (map (*3)) [5,6,7]`
   (c) (0.5) `map (filter (> 3)) [[0..10],[10..0],[]]`
   (d) (0.5) `[3] : []`

2. (2,0) Considere uma aplicação bancária, onde uma lista de pares representa as contas do banco, ou seja, o par ("2287", 532.98) indica que a conta número 2287 tem 532.98 reais de saldo. Defina funções para:
   (a) (0.5) retornar o saldo de todas as contas do banco utilizando foldering.
   `saldoTotal :: Banco -> Float`
   (b) (1.0) realizar crédito, débito e consultar o saldo de uma conta.
   `saldo :: Banco -> Numero -> Saldo`
   `creditar :: Banco -> Numero -> Float -> Banco`
   `debitar :: Banco -> Numero -> Float -> Banco`
   (c) (0.5) dado dois número de conta e um valor realizar uma transferência
   `transferir :: Banco -> Numero -> Numero -> Float -> Banco`

   Dado que
   `type Numero = String`
   `type Saldo = Float`
   `type Banco = [(Numero, Saldo)]`

3. (3.0) Explique a definição da classe Eq abaixo, e defina instâncias de Eq para tuplas e para listas.

```
   class Eq t where
     (==), (/=) :: t -> t -> Bool
     a /= b = not (a==b)
Exemplos:
   (3,4)   == (3,4)
   (1,9)   == (2,1)
   [1,2,1] == [1,2,5]
   []      == []
```

4. (3,0) Prove por indução que `map f (map g x) = (map (f.g) x)` para quaisquer funções `f` e `g` e para todas as listas finitas `x` (indução sobre `x`) dado que.

```
map :: [t] -> (t -> u) -> [u]
map _ [] = []
map f (a:as) = (f a):map f as

(.) :: (u -> v) -> (t -> u) -> (t -> v)
(.) f g x = f (g x)
```