Paradigmas de Programação

Fabrício Olivetti de França

02 de Agosto de 2018

Lista de exercícios 06

1. Defina a instância da `Functor class` para o seguinte tipo de árvores binárias:

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

2. Escreva as instâncias de `Functor` e `Applicative` para o tipo `ZipList`, no qual a função pura faz uma lista infinita de cópias do argumento, e o operador <*> aplica cada função argumento no valor correspondente na mesma posição.

```
newtype ZipList a = Z [a] deriving Show

instance Functor ZipList where
  -- fmap :: (a -> b) -> ZipList a -> ZipList b
  fmap g (Z xs) = ..

instance Applicative ZipList where
  pure :: a -> ZipList a
  pure x = ..
```

4. Dado o tipo

```
data Expr a = Var a | Val Int | Add (Expr a) (Expr a) deriving Show
```

que contém variáveis de um tipo a, defina instâncias para esse tipo de `Functor`, `Applicative` e `Monad`.

5. Defina instâncias de `Functor`, `Applicative` e `Monad` para os seguintes tipos:

```haskell
newtype Identity a = Identity a

data Pair a = Pair a a
```