Lista 07

Programação Funcional

Prof. Maycon Amaro

Exercício 1

Implemente um tipo de dado abstrato genérico para árvore binária e escreva uma instância de `Functor` para ela. Não se esqueça de derivar uma instância de `Show`.

```haskell
data Tree a = Empty | Node (Tree a) a (Tree a)
  -- derive show

instance Functor Tree where
  -- sua solução
```

Exercício 2

Implemente as seguintes funções:

```haskell
incr :: Num a => Tree a -> Tree a
-- incrementa os números da árvore

lowerize :: Tree String -> Tree String
-- transforma todas as strings em lowercase
```

Confira sua implementação com os seguintes exemplos:

```
ghci> incr (Node (Node Empty 5 Empty) 12 (Node Empty (-2) Empty))
Node (Node Empty 6 Empty) 13 (Node Empty (-1) Empty)
ghci> fmap id (Node Empty "Yellow" Empty)
Node Empty "Yellow" Empty
ghci> lowerize (Node Empty "YeLLow" Empty)
Node Empty "yellow" Empty
```

Exercício 3

Implemente a função abaixo, que incrementa todos os números de todas as árvores binárias contextualizadas. Você pode testar sua implementação construindo uma

lista de árvores, ou um `Maybe` de uma árvore ou até mesmo uma tupla em que o segundo elemento é uma árvore. Sua implementação deve funcionar com todos esses casos.

```
incr' :: (Functor f, Num a) => f (Tree a) -> f (Tree a)
```