Exercícios sobre Tipos Algébricos

prof. André Rauber Du Bois

Universidade Federal de Pelotas
dubois@inf.ufpel.edu.br

1 Questionário

1. Defina um tipo algébrico para representar os dias da semana (tipo `Dia`) (construtores: `Segunda`, `Terca`, `Quarta` ...)
2. Defina a função

   ```
   finalDeSemana :: Dia -> Bool
   ```

   que recebe um dia da semana e retorna um booleando dizendo se é final de semana ou não.
3. Defina o tipo algébrico `TalvezFloat` que pode retornar um float, usando o construtor `Valor`, ou retornar uma mensagem de erro com o construtor `Erro`. O construtor erro recebe uma String com a mensagem de erro.
4. Implemente a função

   ```
   divisao :: Float -> Float -> TalvezFloat
   ```

   que recebe dois valores `n1` e `n2` e devolve a divisão de `n1` por `n2` usando o construtor `Valor`. Caso seja uma divisão por zero, a função deve retornar o valor `Erro` com uma mensagem de erro apropriada.
5. Um tipo algébrico pode ser recursivo. Por exemplo, o tipo

   ```
   data Nat = Zero | Suc Nat
      deriving(Eq,Show)
   ```

   define de maneira indutiva uma possível representação dos números naturais. Dessa forma, um número natural pode ser o `Zero`, ou a aplicação do construtor `Suc` (sucessor) em cima de um número natural qualquer. Por exemplo, os números `2` e `7`, poderiam ser representados da seguinte forma:

   ```
   dois :: Nat
   dois = Suc (Suc Zero)


   sete :: Nat
   sete = Suc (Suc (Suc (Suc (Suc (Suc (Suc Zero))))))
   ```

   Defina a função:

```
natToint :: Nat -> Int
```

que transforma um número `Nat` em seu respectivo inteiro

6. Defina a função

```
intToNat :: Int -> Nat
```