Lista 05

Programação Funcional

Prof. Maycon Amaro

Orientações

- Lembre-se que você pode criar tipos de dados e funções auxiliares sempre que julgar necessário ou conveniente.
- Se quiser feedback sobre as suas soluções, envie-a para `maycon.amaro@ufop.edu.br`, iniciando o assunto com [BCC222].

Exercícios

1. Considerando a álgebra de tipos, vamos pensar nos possíveis valores dos tipos $0 \times (1+1)$ e $(0 \times 1) + (0 \times 1)$. Em Haskell, isso seria constatar os possíveis valores de `(Void, Either () ())` e `Either (Void, ()) (Void, ())`. No primeiro caso, não podemos construir a tupla, porque isso exigiria construir um `Void`, logo não há possíveis valores. No segundo caso, independentemente se vamos construir o tipo da esquerda ou o tipo da direita, ambos exigem a construção de `Void`, logo também não há possíveis valores. Isso vai de encontro com a propriedade da distributividade do produto sobre a soma: $A \times (B + C) = (A \times B) + (A \times C)$, que também é verdadeiro na matemática. Já a distributividade da soma sobre o produto não acontece na matemática. Reflita se ela acontece na álgebra de tipos, ou seja, se $A + (B \times C) = (A + B) \times (A + C)$.
2. Defina o tipo `TreeInt` e a função `profundidade` como apresentado em aula e use o `ghci` para testá-la.
3. Defina a função `centralPath :: TreeInt -> [Int]` que constrói uma lista a partir dos elementos da árvore, pela estratégia do caminhamento central. Nesta estratégia, cada nó deve ter sua sub-árvore à esquerda inserida antes de seu próprio elemento, que deve ser inserido antes de sua sub-árvore à direita.
4. Defina a função `dobro :: TreeInt -> TreeInt` que multiplica todos os elementos de uma árvore por 2. Note que não será possível usar funções de ordem superior. Veremos como resolver isso futuramente.

5. Defina o tipo `Pessoa` como apresentado em aula e defina a função `somaIdades :: [Pessoa] -> Int` que retorna a soma das idades das pessoas na lista. Tente usar funções de ordem superior.
6. Defina o tipo `FaixaEtaria` que é uma enumeração das constantes `Criança`, `Jovem`, `Adulto`, `Idoso` e defina a função `faixa :: Pessoa -> FaixaEtaria` que retorna a faixa etária da pessoa. Você pode escolher os limites de idade de cada faixa etária. Note que sua função será parcial (idades negativas não terão resposta).
7. (Um pouco difícil) Defina a função `perfil :: [Pessoa] -> FaixaEtaria` que retorna a faixa etária que mais aparece no grupo de pessoas. Se duas ou mais faixas etárias empatarem, retorne qualquer uma delas.