Programação Funcional

Ficha 7

Árvores binárias com conteúdo nos nós

1. Considere o seguinte tipo para representar árvores binárias.

```
data BTree a = Empty
             | Node a (BTree a) (BTree a)
          deriving Show
```

   Defina as seguintes funções:

   (a) `altura :: BTree a -> Int` que calcula a altura da árvore.

   (b) `contaNodos :: BTree a -> Int` que calcula o número de nodos da árvore.

   (c) `folhas :: BTree a -> Int`, que calcula o número de folhas (i.e., nodos sem descendentes) da árvore.

   (d) `prune :: Int -> BTree a -> BTree a`, que remove de uma árvore todos os elementos a partir de uma determinada profundidade.

   (e) `path :: [Bool] -> BTree a -> [a]`, que dado um caminho (`False` corresponde a *esquerda* e `True` a *direita*) e uma árvore, dá a lista com a informação dos nodos por onde esse caminho passa.

   (f) `mirror :: BTree a -> BTree a`, que dá a árvore simétrica.

   (g) `zipWithBT :: (a -> b -> c) -> BTree a -> BTree b -> BTree c` que generaliza a função `zipWith` para árvores binárias.

   (h) `unzipBT :: BTree (a,b,c) -> (BTree a,BTree b,BTree c)`, que generaliza a função `unzip` (neste caso de triplos) para árvores binárias.

2. Defina as seguintes funções, assumindo agora que as árvores são binárias de procura:

   (a) Defina uma função `minimo :: Ord a => BTree a -> a` que calcula o menor elemento de uma árvore binária de procura **não vazia**.

   (b) Defina uma função `semMinimo :: Ord a => BTree a -> BTree a` que remove o menor elemento de uma árvore binária de procura **não vazia**.

   (c) Defina uma função `minSmin :: Ord a => BTree a -> (a,BTree a)` que calcula, com uma única travessia da árvore o resultado das duas funções anteriores.

   (d) Defina uma função `remove :: Ord a => a -> BTree a -> BTree a` que remove um elemento de uma árvore binária de procura, usando a função anterior.

3. Considere agora que guardamos a informação sobre uma turma de alunos na seguinte estrutura de dados:

```
type Aluno = (Numero,Nome,Regime,Classificacao)
type Numero = Int
type Nome = String
data Regime = ORD | TE | MEL  deriving Show
data Classificacao = Aprov Int
                   | Rep
                   | Faltou
   deriving Show
type Turma = BTree Aluno  -- árvore binária de procura (ordenada por número)
```

Defina as seguintes funções:

(a) `inscNum :: Numero -> Turma -> Bool`, que verifica se um aluno, com um dado número, está inscrito.

(b) `inscNome :: Nome -> Turma -> Bool`, que verifica se um aluno, com um dado nome, está inscrito.

(c) `trabEst :: Turma -> [(Numero,Nome)]`, que lista o número e nome dos alunos trabalhadores-estudantes (ordenados por número).

(d) `nota :: Numero -> Turma -> Maybe Classificacao`, que calcula a classificação de um aluno (se o aluno não estiver inscrito a função deve retornar `Nothing`).

(e) `percFaltas :: Turma -> Float`, que calcula a percentagem de alunos que faltaram à avaliação.

(f) `mediaAprov :: Turma -> Float`, que calcula a média das notas dos alunos que passaram.

(g) `aprovAv :: Turma -> Float`, que calcula o rácio de alunos aprovados por avaliados. Implemente esta função fazendo apenas uma travessia da árvore.