Programação Funcional

Ficha 7

Outros tipos de árvores

1. Considere o seguinte tipo para representar expressões inteiras.

Os termos deste tipo ExpInt podem ser vistos como árvores cujas folhas são inteiros e cujos nodos (não folhas) são operadores.

- (a) Defina uma função calcula :: ExpInt -> Int que, dada uma destas expressões calcula o seu valor.
- (b) Defina uma função infixa :: ExpInt -> String de forma a que infixa (Mais (Const 3) (Menos (Const 2) (Const 5))) dê como resultado "(3 + (2 - 5))".
- (c) Defina uma outra função de conversão para strings posfixa :: ExpInt -> String de forma a que quando aplicada à expressão acima dê como resultado "3 2 5 -+".
- Considere o seguinte tipo para representar árvores irregulares (rose trees).

```
data RTree a = R a [RTree a]
```

Defina as seguintes funções sobre estas árvores:

- (a) soma :: Num a => RTree a -> a que soma os elementos da árvore.
- (b) altura :: RTree a -> Int que calcula a altura da árvore.
- (c) prune :: Int -> RTree a -> RTree a que remove de uma árvore todos os elementos a partir de uma determinada profundidade.
- (d) mirror :: RTree a -> RTree a que gera a árvore simétrica.
- (e) postorder :: RTree a -> [a] que corresponde à travessia postorder da árvore.
- 3. Relembre a definição de árvores binárias apresentada na ficha anterior:

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

Nestas árvores a informação está nos nodos (as extermidades da árvore têm apenas uma marca – Empty). É também habitual definirem-se árvores em que a informação está apenas nas extermidades (leaf trees):

```
data LTree a = Tip a | Fork (LTree a) (LTree a)
```

Defina sobre este tipo as seguintes funções

- (a) ltSum :: Num a => LTree a -> a que soma as folhas de uma árvore.
- (b) listaLT :: LTree a -> [a] que lista as folhas de uma árvore (da esquerda para a direita).
- (c) ltHeight :: LTree a -> Int que calcula a altura de uma árvore.
- 4. Estes dois conceitos podem ser agrupados num só, definindo o seguinte tipo:

```
data FTree a b = Leaf b | No a (FTree a b) (FTree a b)
```

São as chamadas full trees onde a informação está não só nos nodos, como também nas folhas (note que o tipo da informação nos nodos e nas folhas não tem que ser o mesmo).

- (a) Defina a função splitFTree :: FTree a b -> (BTree a, LTree b) que separa uma árvore com informação nos nodos e nas folhas em duas árvores de tipos diferentes.
- (b) Defina ainda a função joinTrees :: BTree a -> LTree b -> Maybe (FTree a b) que sempre que as árvores sejam compatíveis as junta numa só.