Exame de Época Especial de Programação Funcional

1º Ano, MIEI / LCC / MIEF

11 de Setembro de 2020, 9:00 (Duração: 2 horas)

1. Apresente uma definição recursiva das seguintes funções sobre listas:

   (a) `subst :: Eq a => (a,a) -> [a] -> [a]`, que dado um par de elementos e uma lista, calcula a lista que resulta de substituir todas as ocorrências na lista do primeiro elemento do par dado, pelo segundo elemento.
   Por exemplo, `subst (3,0) [1,2,3,4,3,2,3,4,5]` corresponde a `[1,2,0,4,0,2,0,4,5]`.

   (b) `posicoes:: [a] -> [Int] -> [a]` que retorna a lista resultante de selecionar da primeira lista os elementos que estão nas posições indicadas na segunda lista.
   Por exemplo, `posicoes [7,4,9,1,2,3,4,5,1] [1,5,2]` corresponde a `[7,2,4]` .

2. Considere a seguinte definição de árvores binárias com informação (do tipo `a`) nos nodos intermédios e com informação (do tipo `b`) nas folhas

   ```
   data Tree a b = Leaf b | Node a (Tree a b) (Tree a b)
   ```

   (a) Defina a função `folhas :: Tree a b -> [b]`, lista as folhas da árvore (da esquerda para a direita).

   (b) Defina a função `somas :: Tree Float Int -> (Float,Int)`, que calcula os somatórios de todos os nodos intermédios e de todas as folhas da árvore. Implemete esta função fazendo uma única travessia da árvore.

3. Considere a sequinte definição para representar matrizes: `type Mat a = [[a]]`

   Por exemplo, a matriz $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$ seria representada por `[[1,2,3], [0,4,5], [0,0,6]]`

   Defina a função `rotateLeft :: Mat a -> Mat a` que roda uma matriz 90º para a esquerda.

   Por exemplo, o resultado de rodar a matriz acima apresentada deve corresponder à matriz $\begin{bmatrix} 3 & 5 & 6 \\ 2 & 4 & 0 \\ 1 & 0 & 0 \end{bmatrix}$.

4. Considere as seguintes definições de tipos de dados para representar filmes:

```
type Filme = (Titulo,Realizador,[Actor],Genero,Ano)
type Titulo = String
type Realizador = String
type Actor = String
type Ano = Int

data Genero = Comedia | Drama | Ficcao | Accao | Animacao | Documentario
   deriving Eq

type Filmes = [Filme]
```

   (a) Defina a função `doRealizador :: Filmes -> Realizador -> [Titulo]` que lista o nome dos filmes realizados por um dado realizador.

   (b) Defina a função `doActor :: Filmes -> Actor -> [Titulo]` que lista o nome dos filmes em que um dado actor participa.

   (c) Analise a seguinte definição

```
consulta :: Filmes -> Genero -> Realizador -> [(Ano, Titulo)]
consulta bd gen rea = map aux (filter (teste gen rea) bd)
  where teste ::  Genero -> Realizador -> Filme -> Bool
        teste g r (_,x,_,y,_) = g==y && r==x
```

   e apresente a definição de `aux`, incluindo o seu tipo.

5. Pretende-se agora enriquecer a base de dados de filmes, associando a cada filme uma lista de avaliações com classificações do filme. Para isso, definiram-se os seguintes tipos de dados:

```
data Avaliacao = NaoVi
               | Pontos Int        -- pontuação entre 1 e 5

type FilmesAval = [(Filme,[Avaliacao])]
```

   (a) Defina a função `avalia :: FilmesAval -> IO FilmesAval` que lê o título de um filme e a sua avaliação e, caso o filme exista, acrescenta essa avaliação à base de dados de filmes. A escala de classificação deve ser de 1 a 5.

   (b) Defina a função `listaPorGeneros :: FilmesAval -> [(Genero,[(Titulo,Avaliacao)])]` que calcula a avaliação de cada filme, e apresenta o resultado organizado por género. A avaliação deve ser a média das classificações, e `NaoVi` não deve contar para a média. Caso um filme não tenha nenhuma avaliação com pontos, o resultado da média da avaliação desse filme deverá ser `NaoVi`.