Aula Teórico-Prática 3

Programação Funcional

LEI 1º ano

1. Defina recursivamente as seguintes funções sobre listas:

   (a) `dobros :: [Float] -> [Float]` que recebe uma lista e produz a lista em que cada elemento é o dobro do valor correspondente na lista de entrada.

   (b) `ocorre :: Char -> String -> Int` que calcula o número de vezes que um caracter ocorre numa string.

   (c) `pmaior :: Int -> [Int] -> Int` que recebe um inteiro n e uma lista l de inteiros e devolve o primeiro número em l que é maior do que n. Se nenhum número em l for maior do que n, devolve n.

   (d) `repetidos :: [Int] -> Bool` que testa se uma lista tem elementos repetidos.

   (e) `nums :: String -> [Int]` recebe uma string e devolve uma lista com os algarismos que occorem nessa string, pela mesma ordem. (Obs: relembre as funções da Ficha 2).

   (f) `tresUlt :: [a] -> [a]` devolve os últimos três elementos de uma lista, Se a lista de entrada tiver menos de três elementos, devolve a própria lista.

   (g) `posImpares :: [a] -> [a]` calcula a lista com os elementos que occorem nas posiçãoes impares da lista de entrada.

2. Considere o seguinte tipo de dados para armazenar informação sobre uma turma de alunos:

```
type Aluno = (Numero,Nome,ParteI,ParteII)
type Numero = Int
type Nome = String
type ParteI = Float
type ParteII = Float
type Turma = [Aluno]
```

   Defina funções para:

   (a) Testar se uma turma é válida (i.e., os alunos tem todos números diferentes, as notas da Parte I estão entre 0 e 12, e as notas da Parte II entre 0 e 8).

   (b) Seleciona os alunos que passaram (i.e., a nota da Parte I não é inferior a 8, e a soma das nota da Parte I e II é superior ou igual a 9,5).

   (c) Calcula a nota final dos alunos que passaram.

   (d) Calcular a média das notas dos alunos que passaram.

   (e) Determinar o nome do aluno com nota mais alta.

3. Assumindo que uma hora é representada por um par de inteiros, uma viagem pode ser representada por uma sequência de etapas, onde cada etapa é representada por um par de horas (partida, chegada):

```
type Hora = (Int,Int)
type Etapa = (Hora,Hora)
type Viagem = [Etapa]
```

Por exemplo, se uma viagem for

```
[((9,30), (10,25)), ((11,20), (12,45)) , ((13,30), (14,45))]
```

significa que teve três etapas:

- a primeira começou às 9 e um quarto e terminou às 10 e 25;
- a segunda começou às 11 e 20 e terminou à uma menos um quarto;
- a terceira começou às 1 e meia e terminou às 3 menos um quarto;

Utilizando as funções sobre horas que definiu na Ficha 2, defina as seguintes funções:

(a) Testar se uma etapa está bem construída (i.e., o tempo de chegada é superior ao de partida e as horas são válidas).

(b) Testa se uma viagem está bem construída (i.e., se para cada etapa, o tempo de chegada é superior ao de partida, e que a etapa seguinte começa depois de a etapa anterior ter terminado).

(c) Calcular a hora de partida e de chegada de uma dada viagem.

(d) Dada uma viagem válida, calcular o tempo total de viagem efectiva.

(e) Calcular o tempo total de espera.

(f) Calcular o tempo total da viagem (a soma dos tempos de espera e de viagem efectiva).