Princípios de Programação

Trabalho para casa 4

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2019/2020

Neste trabalho pretendemos continuar a desenvolver partes de um sistema de navegação. Vamos contruir um módulo para criar e visualizar rotas. Desta vez fornecemos um módulo chamado `Geometria` que disponibiliza duas abreviaturas de tipo e algumas funções que podem ser úteis para o vosso trabalho. Uma das abreviaturas de tipo representa uma posição geográfica dada pelas suas coordenadas. A outra um percurso dado por uma lista de pontos

```
type Ponto = (Float, Float)
type Percurso = [Ponto]
```

A. Defina o(s) tipo(s) de dados necessários para representar uma *rota*. Cada rota é caracterizada por um nome e uma lista de paragens. Cada paragem pode ser um ponto de interesse—com um nome e um ponto—ou uma paragem técnica—com um ponto apenas. O tipo de dados dever-se-á chamar `Rota`.

B. Escreva a função `criaRota :: String -> [String] -> Percurso -> Rota` que recebe o nome da rota, uma lista de nomes de pontos de interesse e uma lista de pontos. A função devolve uma rota com pontos de interesse apenas (sem paragens técnicas) e com a informação constante nas listas. A dimensão da lista resultante é igual à dimensão da lista mais curta, entre as duas listas dadas.

C. Escreva a função `adicionaTecnica :: Int -> Ponto -> Rota -> Rota` que recebe um índice, um ponto e uma rota. A função acrescenta uma paragem técnica no ponto dado, após o índice. Assuma que o índice denota uma posição válida na rota.

D. Torne o tipo de dados `Rota` uma instância da classe de tipos **`Show`**. A representação textual de uma rota é a a seguinte: nome da rota, seguido de `"␣("`, seguido da distancia do percurso arredondado às unidades, seguido de `"):␣"`, seguido dos nomes das várias paragens separadas por `"␣---␣"`. Os pontos de interesse são representados pelo seu nome; as paragens técnicas são representadas por `"(Pausa)"`. As coordenadas dos pontos não aparecem na representação textual. Sugestão: utilize a função **`round`** para obter o inteiro mais próximo de um número em vírgula flutuante. Atenção aos espaços: vamos testar os vossos trabalhos de um modo automático.

E. Prepare um módulo chamado `Rotas`. O módulo deverá exportar o tipo de dados `Rota` e as funções `criaRota` e `adicionaTecnica`. Um exemplo:

```
*Rotas> let ns = ["Castelo S. Jorge", "Basilica da Estrela",
 "Praca do Comercio"]
*Rotas> let ps = [(0.231, 0.431), (1.312, 4.121), (123.312, 1.0)]
*Rotas> adicionaTecnica 1 (2.332, 5.131) $ criaRota "Historia" ns ps
Historia (129): Castelo S. Jorge --- (Pausa) --- Basilica da
 Estrela --- Praça do Comercio
```

Notas

1. O seu trabalho é constituído pelo módulo `Rotas`. Respeite este nome.
2. Os trabalhos serão avaliados automaticamente. Respeite os nomes e os tipos das *duas* funções `criaRota` e `adicionaTecnica`, bem como do tipo de dados `Rota`.
3. Não se esqueça de apresentar uma assinatura para cada função *top-level* que escrever.
4. Para resolver este exercício pode utilizar toda a matéria do livro de texto até ao capítulo sobre criação de tipos (*Making Our Own Types and Typeclasses*).
5. Pode usar qualquer função constante no **`Prelude`** ou em qualquer outro módulo da biblioteca `Haskell`.
6. Lembre-se que as boas práticas de programação Haskell apontam para a utilização de várias funções simples em lugar de uma função única mas complicada.

Entrega. Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 20 de novembro de 2019.

Ética. Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Lembre-se: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".