Princípios de Programação

Trabalho para casa 1

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

2019/2020

Neste trabalho pretende-se construir partes de um sistema de condução autónoma. Para tal é necessário manipular a posição geográfica de veículos e outros objectos. Vamos representar posições como pontos 2D num mapa. Ao longo deste trabalho, considere um ponto 2D representado pelo tipo `(Float, Float)`.

A. Sabendo que as coordenadas de três pontos estão à mesma distância, neste exercício pretende-se triangular a posição de um objecto. Escreva a função `triangulacao` que recebe um triplo de pontos 2D e devolva um ponto 2D que corresponde ao ponto equidistante dos outros. A triangulação de $((x_1, y_1), (x_2, y_2), (x_3, y_3))$ é:

$$(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3})$$

Um exemplo de utilização é:

```
> triangulacao ((3,4), (3,4), (3,4))
(3.0,4.0)
> triangulacao ((3,4), (1,10), (2,8))
(2.0,7.3333335)
```

B. Escreva a função `distanciaOrigem` que recebe uma lista de pontos 2D e devolva uma lista com as distâncias de cada ponto à origem do referencial. Um exemplo de utilização é:

```
> distanciaOrigem [(1,1),(0,1),(1,0),(0,0),(3,4)]
[1.4142135,1.0,1.0,0.0,5.0]
```

C. Considerando que uma lista de pontos pode ser considerada uma rota de um veículo, neste exercício pretende-se prever a posição seguinte desse veículo. Essa posição prevista é calculada aplicando o vector dado pelos últimos dois pontos da lista ao último ponto.
Se os dois últimos pontos forem Y e Z, o próximo ponto da rota será dado por $Z + (Z - Y)$.
Implemente esta funcionalidade na função `proximoPonto` que recebe uma lista de pelo menos dois pontos 2D e devolve o ponto 2D previsto.
Um exemplo de utilização é:

```
> proximoPonto [(5,6), (0,0), (1,1)]
(2.0,2.0)
```

Notas

1. Os trabalhos serão avaliados automaticamente. Respeite os nomes e os tipos das *três* funções `triangulacao`, `distanciaOrigem` e `proximoPonto`.
2. Não se esqueça de apresentar uma assinatura para cada função que escrever.
3. Para resolver estes problemas deve utilizar *apenas* a matéria dos três primeiros capítulos do livro (de "Starting Out" até "Syntax in Functions"). Pode usar qualquer função constante no **`Prelude`**.
4. 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 9 de outubro 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)".