Aula Prática 3

Instruções:

1 - Os exercícios práticos devem ser realizados individualmente e enviados por e-mail com o assunto [IF686EC] AP 3 para monitoria-if686-ec-l@cin.ufpe.br até as **23:59 de sábado (13.04.2019)**.

2 - As resoluções dos exercícios devem estar em arquivos diferentes, um arquivo por exercício com os nomes no formato Q[número da questão].hs. Nesse caso: Q1.hs, Q2.hs e Q[BONUS].hs

3 - O arquivo com a resposta de cada questão deve conter a função solicitada no formato dado em negrito no enunciado da questão. Os tipos de entrada e saída explicitados, assim como o nome da função, devem ser respeitados.

[Q1] Defina uma função **classificados :: Grupo -> [Jogo] -> (Time, Time)** que, dado um Grupo e uma lista de jogos, retorne o par de times que estão classificados. Os classificados são: os dois com maior número de pontos; em caso de empate, usa-se o saldo de gols; em caso de continuar empate usa-se o número de gols feitos (há regras adicionais, mas vamos implementar apenas essas 3). Exemplos de grupos são: Grupo A: Egito, Russia, Arabia e Uruguai; Grupo B: Ira, Marrocos, Portugal e Espanha.

type Grupo = (Char, Time, Time, Time, Time)

```
data Time = Egito | Russia | Arabia | Uruguai |
            Ira | Marrocos | Portugal | Espanha
            deriving (Show)
```

type Jogo = (Time, Int, Int, Time)

Por exemplo: (Egito 3 x 1 Russia) será representado por (Egito, 3, 1, Russia) e o Grupo A seria ('A', Egito, Russia, Arabia, Uruguai) e o Grupo B seria ('B', Ira, Marrocos, Portugal, Espanha]

```
jogos1 :: [Jogo]
jogos1 = [(Egito, 1, 3, Russia), (Arabia, 0, 3, Uruguai),
          (Egito, 0, 0, Arabia),(Russia, 0, 2, Uruguai),
          (Russia, 2, 0, Arabia), (Egito, 0, 2, Uruguai),
          (Ira, 1, 1, Marrocos), (Portugal, 2, 2, Espanha),
          (Ira, 1, 2, Portugal), (Ira, 0, 1, Espanha),
          (Marrocos, 0, 3, Portugal), (Marrocos, 1, 1, Espanha)]
```

[Q2] Faça uma função **`destination :: (Int,Int) -> [Command] -> (Int,Int)`** que informe a localização do robô após uma sequêcia de comandos, supondo que o robô comece na posição (0,0) (coordenadas) e direcionado para norte (i.e. para o posição (0,1)). Um robô é controlado por 4 comandos:

Left, para girar sua direção à esquerda 90 graus;
Right, para girar sua direção à direita em 90 graus;
Forward seguido de um número N, que indica um avanço de N metros.
Backward seguido de um número N, que indica um retrocesso de N metros.

Exemplo de posições/coordenadas:
(-2, 2) (-1, 2) (0, 2) (1, 2) (2, 2)
(-2, 1) (-1, 1) (0, 1) (1, 1) (2, 1)
(-2, 0) (-1, 0) (0, 0) (1, 0) (2, 0)
(-2,-1) (-1,-1) (0,-1) (1,-1) (2,-1)
(-2,-2) (-1,-2) (0,-2) (1,-2) (2,-2)

```
data Command = Forward Int | Backward Int | TurnLeft | TurnRight
               deriving (Eq, Show)

data Direction = ToNorth | ToSouth | ToWest | ToEast

Exemplo:
destination (0,0) [Forward 2, TurnLeft, TurnLeft, Forward 1]
> (0,1)
destination (0,0) [Backward 2, Forward 1]
> (0,-1)
```

Q[BONUS] Faça uma função **`faces :: Direction -> [Command] -> Direction`** que informe para qual direção o robô estará voltado ao final de uma sequência de comandos (ToNorth, ToSouth, ToEast ou ToWest), assumindo que ele começa voltado para a direção ToNorth.

```
exemplo: faces ToNorth [Forward 2, TurnLeft, TurnLeft, Forward 1]
---> ToSouth
         faces ToNorth [Backward 2, Forward 1] ---> North
         faces ToNorth [TurnLeft, TurnLeft, TurnLeft] ---> ToEast
```