# Aula Prática 2

**Instruções:**

1 - Os exercícios práticos devem ser realizados individualmente e enviados por e-mail com o assunto [IF686EC] AP 2 para **monitoria-if686-ec-l@cin.ufpe.br** até as **23:59 do segunda (09.09.2019)**.

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, são esperados 5 possíveis arquivos: Q1.hs, Q2.hs, Q3.hs, Q4.hs e Q5.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, devem ser respeitados, assim como o nome da função.

---

Data;Tipo;Compra;Valor;
14 JAN;Amazon;40.32;
15 JAN;Uber;14.84;
25 JAN;Uber;34.24;
02 FEV;Spotify;8.50;
06 FEV;Uber;6.94;
05 MAR;Burger;29.90;
10 MAR;Burger;24.99;
15 MAR;UCI;19.00;
08 ABR;Itunes;3.50;
13 ABR;Picpay;20.00;

---

Considere a fatura anual de cartão acima, sendo representada pela String:
logCartao = "14 JAN;Amazon;40.32;15 JAN;Uber;14.84;25 JAN;Uber;34.24;02 FEV;Spotify;8.50;06 FEV;Uber;6.94;05 MAR;Burger;29.90;10 MAR;Burger;24.99;15 MAR;UCI;19.00;08 ABR;Itunes;3.50;13 ABR;Picpay;20.00;"

**[Q1]** Escreva uma função **logMes :: String -> String -> Double** que recebe uma String (JAN, FEV, MAR ou ABR), uma String referente a fatura anual e retorna o total gasto no mês em questão.

```
logMes "JAN" logCartao
89.4
```

**[Q2]** Escreva uma função **minMaxCartao :: String -> (Double, Double)** que recebe uma String referente a fatura anual e retorna uma tupla com o menor e o maior dos valores gastos.

```
minMaxCartao logCartao
(3.5, 40.32)
```

**[Q3]** Escreva uma função **processBankOperations :: [Double] -> [(Int, Int, Int, Double)] -> [Double]** que recebe uma lista com os saldos iniciais de contas bancárias e uma lista de operações bancárias e retorna uma lista com os saldos atualizados das contas bancárias após as operações.

As operações são fornecidas como uma tupla no seguinte formato:

(**OpCode**, **ContaOrigem**, **ContaDestino**, **Valor**)

Onde OpCode pode ser:

0: Crédito de **Valor** em **ContaOrigem**

1: Débito de **Valor** em **ContaOrigem**

2: Transferência de **Valor** da **ContaOrigem** para **ContaDestino**

**ContaOrigem** e **ContaDestino** se referem ao índice da conta bancária na lista fornecida.

Caso a **ContaOrigem** não tenha saldo para realizar uma operação, esta deverá ser ignorada.

```
processBankOperations [150.0, 50.0] [(1, 1, 0, 100.0), (2, 0, 1, 50.0), (0, 1, 0, 25.0)]
[100.0, 125.0]
```

**[Q4]** Faça uma função `isEqTriangle :: [(Double, Double)] -> Bool` que recebe uma lista de tuplas representando coordenadas cartesianas (x, y) de pontos e retorna se esses pontos representam um triângulo equilátero.

Exemplo:

```
isEqTriangle [(0,0),(1,0),(0.5,sqrt 0.75)]
True
isEqTriangle [(0,0),(1,0),(0,1)]
False
```

**[Q5]** Crie uma função `executa :: [(Comando, Valor)] -> Int` que recebe uma lista de tuplas com comandos e valores. A função deve começar pelo valor. Por exemplo, caso a sequência de comandos seja [("Multiplica", 2), ("Soma", 5), ("Subtrai", 3)], a função deve pegar 0 e efetuar as seguintes operações: (((0 * 2) + 5) -3). Esses comandos podem ser "Multiplica", "Soma", "Subtrai" ou "Divide". Para o caso de uma divisão por 0, a função deve parar retornando o valor -666 independente de quanto tenha calculado até essa divisão ou de quantos comandos ainda restarem.

```
type Comando = String
type Valor = Int
```

```
executa [("Multiplica", 2), ("Soma", 5), ("Subtrai", 3)]
2
executa [("Multiplica", 2)]
0
executa [("Multiplica", 2), ("Soma", 5), ("Subtrai", 3), ("Soma",
6)]
8
executa [("Multiplica", 2), ("Soma", 5), ("Divide", 0)]
-666
```