**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 quarta-feira (03.10.2018).

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

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

**1)** Dado o tipo de dados *Tree t* abaixo, que representa uma árvore binária com informações (valores) em seus nós, faça uma função **isSortedTree :: Ord t => Tree t -> Bool** que informa se uma árvore está ordenada, ou seja, os valores em nós ou folhas na subárvore à esquerda são sempre menores ou iguais ao valor do nó, e os da sub-árvore à direita sempre maiores.

```
data Tree t = Node t (Tree t) (Tree t)
            | Leaf t
```

**Exemplos:**

```
Main> let ex2 = Node 10 (Node 5 (Leaf 3) (Leaf 6)) (Node 15 (Leaf 16) (Leaf 17)) :: Tree Int
Main> let ex1 = Node 10 (Node 5 (Leaf 3) (Leaf 6)) (Node 15 (Leaf 14) (Leaf 17)) :: Tree Int
Main> isSortedTree ex1
True
Main> isSortedTree ex2
False
```

**2)** Faça o que se pede nos quesitos abaixo.

a) Defina um tipo algébrico **UnidadeDeMedida** com 3 construtores (**Quilometros**, **Metros** e **Centimetros**) que terão valores (*Float*) representando medida nas escalas indicadas.

b) Crie instâncias das classes *Ord*, *Eq* e *Show* para **UnidadeDeMedida**, levando em conta que (MedidaEmQuilometros) = (MedidaEmCentimentros/100000) = (MedidaEmMetros/1000).

c) Crie uma função **minMaxMedidas**, que recebe uma lista medidas e retorna um par em que o primeiro elemento é a menor medição da lista e o segundo elemento a maior.

d) Qual a diferença entre avaliação estrita e preguiçosa (lazy)? Mostre exemplos desta diferença.

**[BÔNUS]** Utilizando a árvore da questão 1, construa uma função **alturaArvore :: Tree t - > Int** que retorne a altura de uma árvore dada. Lembrando que, a altura da árvore é definida por sua sub-árvore mais "alta".

**Exemplos:**

```
Main> let ex1 = Node 10 (Node 5 (Leaf 3) (Leaf 6)) (Node 15 (Leaf 16) (Leaf 17)) :: Tree Int
Main> alturaArvore ex1
3
Main> alturaArvore (Leaf 4)
1
```