

## Aula Prática 2

### Instruções:

1 - Os exercícios práticos devem ser realizados individualmente e enviados por e-mail com o assunto **[IF686EC] AP2** para [monitoria-if686-ec-l@cin.ufpe.br](mailto:monitoria-if686-ec-l@cin.ufpe.br) até as **23:59 de sexta-feira (5.04.2019)**.

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 Q3.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]** (25%) Dados os seguintes tipos algébricos

```
data Ops = SUM | SUB | MUL
```

```
data IntTree = Nilt Int |  
              Node Ops IntTree IntTree
```

Escreva uma função **evalTree :: IntTree -> Int** que calcula o valor resultante das operações na árvore dada.

Exemplo:

```
evalTree (Node SUM (Node MUL (Nilt 5) (Nilt 3)) (Node SUB (Nilt 10)  
(Nilt 5)))  
20
```

**[Q2]** (50%) Dado o seguinte tipo algébrico

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

Escreva uma função **isBST :: Tree t -> Bool** que checa se uma árvore é uma árvore de busca binária. Considerar que nenhum dos elementos se repetirá.

Como informamos via e-mail, **isBST :: Tree t -> Bool** deve ser **isBST :: (Ord t) => Tree t -> Bool** ou **isBST :: IntTree -> Bool** considerando que a árvore receberá

apenas números inteiros. Para o segundo caso, será necessário alterar também o tipo da árvore, passando a ser da seguinte forma:

```
data IntTree = Nilt  
              | Node Int (IntTree) (IntTree)
```

Exemplos:

```
isBST (Node 5 (Node 3 Nilt Nilt) (Node 7 Nilt Nilt))  
True  
isBST (Node 3 (Node 5 Nilt Nilt) (Node 7 Nilt Nilt))  
False
```

**[Q3]** (25%) Escreva a função `mapList :: (t -> t) -> List t -> List t` que recebe uma função e uma lista e retorne o resultado da aplicação dessa função sobre a lista dada.

```
data List t = Nil |  
             Cons t (List t)
```

Exemplo:

```
sumAll :: List t -> t  
mapList (*2) (Cons 3 (Cons 2 Nil))  
Cons 6 (Cons 4 Nil)
```

Para essa lista, são esperados 4 arquivos. São eles: **Q1.hs**, **Q2.hs**, e **Q3.hs**.  
Favor não enviar arquivos com nomes diferente desses.