# Exercícios de Programação Haskell I

prof. André Rauber Du Bois

Universidade Federal de Pelotas
dubois@inf.ufpel.edu.br

## 1 Questionário

1. Escreva a função `osQuatroSaoIguais` que possui tipo

   ```
   Int -> Int -> Int -> Int -> Bool
   ```

   que retorna `True` se seus quatro argumentos são iguais
2. Defina a função `quantosSaoIguais :: Int -> Int -> Int -> Int` que recebe 3 valores e diz quantos desses valores são iguais
3. Defina a função

   ```
   todosDiferentes :: Int -> Int -> Int -> Bool
   ```

   que retorna `True` se todos os seus argumentos são diferentes. Obs: `m /= n` retorna `True` se `m` e `n` são diferentes
4. O que está errado com a seguinte definição de `todosDiferentes`:

   ```
   todosDiferentes n m p = ( ( n/=m ) && ( m/=p ) )
   ```

5. Escreva uma definição de `quantosSaoIguais` que use a função `todosDiferentes` e a função `todosIguais`
6. Defina a função `elevadoDois :: Int -> Int` que recebe um argumento `n` e devolve como resposta $n^2$
7. Defina a função `elevadoQuatro :: Int -> Int` que recebe um argumento `n` e devolve como resposta $n^4$. Use `elevadoDois` para definir `elevadoQuatro`
8. Supondo que exista uma função `vendas`:

   ```
   vendas :: Int -> Int
   ```

   que devolve a venda semanal de uma loja (ex: `vendas 0` devolve as vendas na semana 0, `vendas 1` devolve as vendas na semana 1, etc. Implemente uma função chamada `vendaTotal`, que recebe um argumento `n` e calcula todas as vendas da semana `0` até a semana `n`. Observe que essa função deve ser recursiva. Exemplo de calculo: As vendas da semana 0 até a semana 2, podem ser calculados usando a seguinte formula:

   ```
   vendas 0 + vendas 1 + vendas 2
   ```