Centro de Informática - UFPE
Disciplina: IF686 - Paradigmas de Linguagens Computacionais (CC)
Lista de exercícios
Docentes: Márcio Cornélio

---

1. Determine se a expressão **map**.(.) está correta. Em caso negativo, justifique. Em caso positivo, indique o tipo resultante e justifique (derive) como você o determinou.

2. Defina uma função sublistas :: [a] -> [[a]] que retorna todas as sublistas de uma lista dada como argumento.

3. Considere uma função polinomial de grau 2 ($f(x) = ax^2 + bx + c$), onde a, b e c são os coeficientes do polinômio.
   (a) Defina a função poli :: **Integer** -> **Integer** -> **Integer** -> **Integer** -> **Integer** que recebe como argumentos os coeficientes de uma função polinomial de grau 2 e devolve uma função de inteiro para inteiro (um polinômio)
   (b) Defina a função listaPoli :: [(**Integer**,**Integer**,**Integer**)] -> [**Integer**->**Integer**] que aguarda uma lista de triplas de inteiros (coeficientes de um polinômio de segundo grau) e devolve uma lista de funções de inteiro para inteiro (polinômios) .
   (c) Defina a função appListaPoli :: [**Integer**->**Integer**] -> [**Integer**] -> [**Integer**] que recebe uma lista de funções de polinômios e uma lista de inteiros. Esta função devolve uma lista de inteiros que resultam da aplicação de cada polinômio da primeira lista aplicada ao inteiro correspondente na segunda lista.

4. Dada uma matriz representada por uma lista de listas, defina funções para:
   (a) indicar se a mesma é uma matriz (se todas as linhas têm o mesmo tamanho)
   (b) permutar a posição de duas linhas x e y, assumindo que x < y. Dica: pode-se utilizar as funções **init**, **take**, **drop** e !!.

5. Dados o tipo algébrico Voto e os tipos Urna e Apuracao
   (a) Estabeleça explicitamente que o tipo Voto é uma instância da classe **Eq** de tal forma que dois votos são iguais se os códigos forem iguais, considerando-se o mesmo cargo em disputa
   (b) Defina uma função que retorna o total de votos de um candidato.
   (c) Defina uma função que retorna o total de votos de cada candidato, ou seja, uma apuração.

```
type Codigo = Int
data Voto = Presidente Codigo | Senador Codigo | Deputado Codigo
          | Branco  deriving (Show)
type Urna = [Voto]
type Apuracao = [(Voto, Int)]

totalVotos :: Urna -> Voto  -> Int
apurar :: Urna -> Apuracao
```

```
(.) :: (b-> c) -> (a -> b) -> a -> c
map :: (a->b)->[a]->[b]

zip [1,2,3] [6,7] = [(1,6),(2,7)]

init [1,2,3] = [1,2]
take 2 [1,2,3] = [1,2]
drop 2 [1,2,3] = [3]
[1,2,3,4,5] !! 2 = 3
```