Centro de Informática - UFPE
IF686 -Paradigmas de Linguagens Computacionais

---

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. Implemente a função `filtrarEInserir :: [[Int]] -> Int -> ([[Int]], Int)` que retorna uma tupla. O primeiro elemento da tupla é constituído de listas de inteiros tais que a soma dos números ímpares é maior que a soma dos números pares. O segundo elemento consiste no produto entre o segundo argumento da função `filtrarEInserir` e a multiplicação da maior soma obtida das listas retornadas. Utilize obrigatoriamente `filter`.

   > filtrarEInserir [[2,3,4,5,6], [1, 2, 3], [9]] 5
   ([[1,2,3], [9]], 45)

   > filtrarEInserir [[2,3,4,5], []] 7
   ([[2,3,4,5]],98)

   > filtrarEInserir [] 5
   ([],0)
6. Defina a função `altMap :: (a -> b) -> (a -> b) -> [a] -> [b]` que, de forma alternada, aplica as duas funções dadas como argumentos aos elementos sucessivos na lista, respeitando a ordem deles.

   > altMap (+10) (+100) [0, 1, 2, 3, 4]

   [10, 101, 12, 103, 14]
7. 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
```