Centro de Informática - UFPE
Primeiro Exercício Escolar IF686 (2019.1)
Data: 09/04/2019
Docente: Márcio Lopes Cornélio
Discente:

Turma: E5
Horário: 10h-12h

Instruções para entrega: Enviar dois arquivos para mlc2@cin.ufpe.br : (1) arquivo .hs, com todas as respostas (copiar e colar os códigos das respostas) e (2) um arquivo no formato pdf com todas as respostas. Escrever o nome na primeira linha dos arquivos. O assunto do email deve estar no formato: "Nome completo 1ee_plc_2019_1"

1. A função VENDAS :: INT -> Int retorna a quantidade semanal de vendas de uma loja. As semanas são numeradas em uma sequência 0, 1, 2, ... Implemente a função zeroVendas que se comporta da seguinte maneira: dado um número n, que assumimos como não negativo, retorna o número de semanas na faixa 0, 1, ..., n em que a quantidade de itens vendidos foi 0 (zero). Implemente definições de zeroVendas

(1,5) (a) Usando compreensão de lista e a função length

(1,5) (b) Usando qualquer função padrão de Haskell, mas sem definir função recursiva, não usando foldr ou foldl, e sem usar compreensão de lista

(1,5) (c) Usando foldr, a lista [0..n] e sem uso de qualquer outra função recursiva. Pode ser necessária uma função auxiliar.

(1,0) 2. (a) Defina um tipo algébrico (e tipos auxiliares que você achar necessário) para representar um bilhete de passagem. Um bilhete pode ser um destes

- um bilhete de trem de uma cidade para uma cidade, que pode ser de primeira classe ou segunda classe
- um bilhete de ônibus de uma cidade para uma cidade
- um bilhete aéreo de uma cidade para uma cidade, que pode ser de classe econômica ou executiva

Cidades podem ser representadas por String

(1,5) (b) Para qualquer bilhete, a primeira cidade é chamada de cidade de origem; a segunda, de destino. Nós representamos uma viagem por uma lista de bilhetes

Uma viagem é válida se para quaisquer bilhetes consecutivos na lista, a cidade de destino do primeiro bilhete é a de partida para o segundo bilhete. Defina a função

```
VALIDA :: [BILHETE] -> BOOL
```

que determina se uma viagem é válida. Assuma que a lista não é vazia.

3. Dado o tipo algébrico NAT

```
data NAT = ZERO | SUCC NAT deriving (Eq, Show)
```

Defina:

(0,5) (a) Uma função que converte números inteiros em números naturais.

(0,5) (b) Uma função que converte números naturais em números inteiros

(1,0) (c) Defina a função soma que soma dois números naturais

(1,0) (d) Defina a função mult que multiplica dois números naturais.

```
(.) :: (b-> c) -> (a -> b) -> a -> c
map :: (a->b)->[a]->[b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr1 :: (a -> a -> a) -> [a] -> a
unzip :: [(a, b)] -> ([a], [b])
```