IME-USP - Programação functional pura com aplicações

1 - Atividade – 13 de janeiro de 2024

INSTRUÇÕES

1. Atividade em trio.
2. Atividade deve ser entregue em um único arquivo compactado ou link de um repositório.
3. O arquivo ou o link do repositório deve ser enviado para o e-mail: felipe.cannarozzo@ime.usp.br
4. O fonte desenvolvido deverá ser apenas na linguagem Haskell.
5. A nota da ativdade vai de zero a dez..
6. Esta lista tem um total de 60 porcento da nota final.

Data limite para entrega: 28/01/2024.

1. (valor 3 pontos)

   1.1 Construa o list comprehension que gere:

   (a) `[(0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]`

   (b) Uma lista de 'a' a 'z' sem as vogais.

   (c) Uma lista de 0 a 50 sem os números 2, 7, 13, 35 e 42

   (d) Uma lista com todas as coordenadas dde um tabuleiro de damas 8x8:
   `([('a',1),('a',2),('a',3) ... ('h',7), ('h',8)])`

   1.2 Implemente as seguintes funções:

   (a) Crie uma função que verifique se o tamanho de uma String é par ou não. Use Bool como retorno.

   (b) Escreva uma função que receba um vetor de Strings e retorne uma lista com todos os elementos em ordem reversa

   (c) Escreva a função head como composição de duas outras.

   (d) Implemente uma função que receba um tipo Int e retorne a conversão dele para binário. A saida da função deve ter um [Int] ou [String].

2. (valor 3 pontos)

2.1 Faça um novo tipo chamado Mes , que possui como valores todos os meses do ano. Implemente:

- A função checaFim , que retorna o número de dias que cada mês possui (considere fevereiro tendo 28 dias).
- A função prox , que recebe um mês atual e retorna o próximo mês.
- A função estacao , que retorna a estação do ano de acordo com o mês e com o hemisfério. (Use apenas tipos criados pela palavra data aqui.)

2.2 Faça o tipo Cripto que possua dois values constructors Mensagem e Cifrado , ambos com um campo String e um value constructor Erro . Faça as funções encriptar e decriptar , seguindo cada exemplo a seguir

```
Prelude>encriptar (Mensagem "FATEC")
Cifrado "GBUFD"
Prelude>decriptar (Cifrado "DBTB")
Mensagem "CASA"
```

Veja que a encriptação deve empurrar cada letra a frente e a decriptação faz o inverso, empurrando uma letra para trás. Use as funções succ e pred , e também list compreeshions. Não é possível encriptar mensagens cifradas e decriptar mensagens

3. (valor 2 pontos)

3.1 Implemente uma função que filtre os números pares e outra que filtre os ímpares de uma lista recebida via parâmetro.

3.2 Implemente o tipo Dinheiro que contenha os campos valor e correncia (Real ou Dolar), e uma função que converta todos os "dinheiros"de uma lista para dólar (e outra para real). Com isso, implemente funções para:

(a) Filtrar todos os Dolares de uma lista de Dinheiro .

(b) Somar todos os Dolares de uma lista.

(c) Aumentar a quantidade o valor dos reais em uma [Dinheiro].

4. (valor 2)

4.1 Dado o tipo de dado:

```
data Lista a = Nulo | a :>: (Lista a)  derivinh Show
```

Implemente a função `removerElemento` que recebe um elemento "a"qualquer, uma Lista a e retorne uma Lista a com o elemento removido. `removerElemento :: a -> Lista a -> Lista a`

4.2 Crie o tipo Paridade com os values constructors Par e Impar .Crie o typeclass ParImpar que contém a função `" decide :: a ->  Paridade "` e possui as instâncias:

- Para Int : noção de Par/Impar de Int .
- Para [a] : uma lista de elementos qualquer é Par se o número de elementos o for.
- Bool : False como Par , True como Impar.

,
4.3 Crie o tipo TipoProduto que possui os values constructors Escritorio , Informatica , Livro , Filme e Total . O tipo Produto possui um value constructor - de mesmo nome - e os campos valor (Double), tp (TipoProduto) e um value constructor Nada , que representa a ausência de um Produto .

Deseja-se calcular o valor total de uma compra, de modo a não ter nenhuma conversão para inteiro e de forma combinável. Crie uma instância de semigrupo e monoide para Produto , de modo que o retorno sempre tenha Total no campo tp e a soma dos dois produtos m valor . Explique como seria o exercício sem o uso de monoides. Qual(is) seria(m) a(s) diferença(s)?

4.4 Dado o tipo de dados

```
data Arvore a = Galho a (Arvore a) (Arvore a) | Folha a | Nulo deriving show
```

- Implemente os percursos pós-ordem e pré-ordem. Via comentário, faça os "testes de mesa"para os dois percursos da árvore Raiz 15 (Raiz 11 (Folha 6) (Raiz 12 (Folha 10) Nula)) (Raiz 20 Nula (Raiz 22 (Folha 21) Nula))
- Usando a estrutura de árvore vista, faça uma função que some todos os elementos de uma árvore de números.

5. (Bônus 2 pontos) Dado o tipo e as implementações:

```
    module Rec where

data Nat = Z | Suc Nat deriving Show

natToInt :: Nat -> Int
natToInt Z = 0
natToInt (Suc n) = 1 + natToInt n

-- input: natToInt (Suc (Suc (Suc (Suc Z))))
-- output: 4

somar :: Nat -> Nat -> Nat
somar x Z = x
somar x (Suc n) = Suc (somar x n)

-- input: somar (Suc (Suc Z)) (Suc (Suc (Suc Z)))
-- output: Suc (Suc (Suc (Suc (Suc Z))))

mult :: Nat -> Nat -> Nat
mult x Z = Z
mult x (Suc Z) = x
mult x (Suc n) = somar x (mult x n)

-- input: mult (Suc (Suc Z)) (Suc (Suc (Suc Z)))
-- output: Suc (Suc (Suc (Suc (Suc Z))))

fat :: Int -> Int
fat n
    | n <= 0 = 1
    | otherwise = n * fat (n - 1)

fatt :: Nat -> Nat
fatt Z = Suc Z
fatt (Suc n) = mult (Suc n) (fatt n)
```

Implemente a função `fibb :: Nat -> Nat` .