Programação Funcional – 1º Ano, LEI / LCC / LEF – 30 de Janeiro de 2023 **Número:** _____ **Nome:** _____ **Curso:** _____

1. Considere que se usa o tipo `type MSet a = [(a,Int)]` para representar multi-conjuntos de elementos de a. Considere ainda que nestas listas não há pares cuja primeira componente coincida, nem cuja segunda componente seja menor ou igual a zero.

   (a) Defina a função `converteMSet :: MSet a -> [a]` que converte um multi-conjunto na lista dos seus elementos Por exemplo, `converteMSet [('b',2), ('a',4), ('c',1)]` corresponde a `"bbaaaac"`.

   (b) Defina a função `removeMSet :: Eq a => a -> MSet a -> MSet a` que remove um elemento a um multi-conjunto. Se o elemento não existir, deve ser retornado o multi-conjunto recebido.

   Por exemplo, `removeMSet 'c' [('b',2), ('a',4), ('c',1)]` corresponde a `[('b',2), ('a',4)]`, e `removeMSet 'a' [('b',2), ('a',4), ('c',1)]` corresponde a `[('b',2), ('a',3), ('c',1)]`.

   (c) Defina a função `uniaoMSet :: Eq a => MSet a -> MSet a -> MSet a` que faz a união de dois multi-conjuntos.

   Por exemplo, `uniaoMSet [('b',2),('a',4),('c',1)] [('c',7),('a',3),('d',5)]` corresponde a `[('c',8),('a',7),('d',5),('b',2)]`

2. Considere o seguinte tipo usado para descrever movimentos de um robot e a sua posição numa grelha.

```
type Posicao = (Int,Int)
data Movimento = Norte | Sul | Este | Oeste
data Caminho = C Posicao  [Movimento]
```

   Defina uma instância da classe `Eq` para o tipo `Caminho`, considerando iguais os caminhos com a mesma posição de partida e de chegada e com o mesmo número de movimentos.

3. Apresente uma definição alternativa da função `func`, usando recursividade explícita em vez de funções de ordem superior e fazendo uma única travessia da lista.

```
func :: [[Int]] -> [Int]
func l = concat (filter (\x -> sum x >10) l)
```

4. Considere o seguinte tipo para representar expressões proposicionais:

```
data Prop = Var String | Not Prop | And Prop Prop | Or Prop Prop

p1 :: Prop
p1 = Not (Or (And (Not (Var "A")) (Var "B")) (Var "C"))
```

   (a) Defina a função `eval :: [(String,Bool)] -> Prop -> Bool` que, dado o valor lógico das variáveis proposicionais, calcula o valor lógico de uma expressão proposicional.

   (b) Uma proposição diz-se na *forma normal negativa* se as negações só estão aplicadas às variáves proposicionais. Por exemplo, a proposição $((A \vee \neg B) \wedge \neg C)$ está na forma normal negativa, enquanto `p1` não está.

   Defina a função `nnf :: Prop -> Prop` que recebe uma proposição e produz uma outra que lhe é equivalente, mas que está na forma normal negativa. Por exemplo, o resultado de `nnf p1` deverá ser a proposição $((A \vee \neg B) \wedge \neg C)$.

   Lembre-se das seguintes leis: $\neg\neg A = A$, $\neg(A \vee B) = \neg A \wedge \neg B$ e $\neg(A \wedge B) = \neg A \vee \neg B$.