Programação Funcional – 1º Ano, LEI / LCC / LEF – 4 de Janeiro de 2024

Número: _____ **Nome:** _____ **Curso:** _____

1. Defina a função `alterna :: Num a => Int -> a -> [a]` que dado um número `n` e um valor `v`, constroí uma lista com `n` elementos alternadamente `v` e `-v`. Por exemplo, `alterna 5 2` corresponde a `[2,-2,2,-2,2]`.

2. Considere o seguinte tipo de dados para guardar os números e nomes dos alunos numa árvore binária de procura ordenada por número de aluno.

```
data Turma = Empty | Node (Integer,String) Turma Turma
```

   (a) Declare `Turma` como instância da classe `Show` de forma a que a visualização da turma seja uma listagem da turma por ordem crescente de número de aluno, com um registo por linha.

   (b) Defina a função `limites :: Turma -> (Integer,Integer)` que dá o par com o menor e o maior número da turma.

3. Considere as seguintes definições de tipos para representar uma tabela de abreviaturas que associa a cada palavra uma abreviatura.

```
type TabAbrev = [(Palavra,Abreviatura)]
type Palavra = String
type Abreviatura = String
```

   (a) Defina a função `difMaior :: TabAbrev -> (Palavra,Int)` que recebe um tabela não vazia, e descobre a palavra da tabela que mais se simplifica (isto é, cuja diferença entre o número de caracteres da palavra e da abreviatura é maior). A função devolve o par com a palavra e o número de caracteres que foi reduzio.
   Por exemplo, `difMaior [("muito","mt"), ("que","q")] == ("muito",3)`.

   (b) Defina a função `subst :: [String] -> TabAbrev -> [String]` que recebe um texto (dado como uma lista de strings) e uma tabela de abreviaturas, substitui todas as abreviaturas que apareçam no texto pelas respectivas palavras associadas.

4. Considere a seguinte definição da função `dumpLT` que, dada uma árvore de folhas, constrói a lista dos seus elementos anotados com o nível em que aparecem na árvore.
   Por exemplo, `dumpLT (Fork (Tip 'a') (Fork (Tip 'b') (Tip 'c')))` corresponde à lista `[('a',2), ('b',3), ('c',3)]`.

```
data LTree a = Tip a | Fork (LTree a) (LTree a)

dumpLT :: LTree a -> [(a,Int)]
dumpLT (Tip x) = [(x,1)]
dumpLT (Fork e d) = map f (dumpLT e  ++ dumpLT d)
    where f :: (a,Int) -> (a,Int)
          f (x,y) = (x,y+1)
```

   (a) Apresente uma definição alternativa, mais eficiente e usando um parâmetro que corresponde ao nível da árvore.

   (b) Defina a função `unDumpLT :: [(a,Int)] -> LTree a`, inversa da anterior, no sentido em que, para toda a árvore `lt` se verifica que `unDumpLT (dumpLT lt) == lt`.