Teste Final de Programação Funcional – 1º Ano, MIEI / LCC / MIEF
10 de Janeiro de 2018 (Duração: 2 horas)

1. Apresente uma definição recursiva da função (pré-definida) `insert :: Ord a => a -> [a] -> [a]` que dado um elemento e uma lista ordenada retorna a lista resultante de inserir ordenadamente esse elemento na lista.

   Por exemplo, `insert 25 [1,20,30,40]` corresponde a `[1,20,25,30,40]`.

2. Apresente uma definição recursiva da função pré-definida `catMaybes :: [Maybe a] -> [a]` que colecciona os elementos do tipo `a` de uma lista.

3. Considere o tipo ao lado para representar expressões aritméticas com variáveis.
   Defina `Exp a` como instância da classe `Show`, de forma a que `show (Mais (Var "x") (Mult (Const 3) (Const 4)))` seja a string `"(x + (3 * 4))"`.

```
data Exp a = Const a
           | Var   String
           | Mais (Exp a) (Exp a)
           | Mult (Exp a) (Exp a)
```

4. Apresente uma definição da função `sortOn :: Ord b => (a -> b) -> [a] -> [a]` que ordena uma lista comparando os resultados de aplicar uma função de extracção de uma chave a cada elemento de uma lista. Por exemplo: `sortOn fst [(3,1),(1,2),(2,5)] == [(1,2),(2,5),(3,1)]`.

5. A amplitude de uma lista de inteiros define-se como a diferença entre o maior e o menor dos elementos da lista (a amplitude de uma lista vazia é 0).

   (a) Defina a função `amplitude :: [Int] -> Int` que calcula a amplitude de uma lista (idealmente numa única passagem pela lista).

   (b) Defina a função `parte :: [Int] -> ([Int],[Int])` que parte uma lista de inteiros em duas, minimizando a soma das amplitudes. Por exemplo, `parte [1,18,3,19,17,20]` deve colocar 1 e 3 numa das listas e os restantes na outra.
   Admita, caso necessite, que existe pré-definida uma função `sort :: Ord a => [a] -> [a]` de ordenação de listas.

6. Considere o seguinte tipo para representar imagens compostas por quadrados (apenas com coordenadas positivas)

```
data Imagem = Quadrado Int
            | Mover (Int,Int) Imagem
            | Juntar [Imagem]
```

   Por exemplo, a seguinte imagem é constituída por três quadrados.

```
ex :: Imagem
ex = Mover (5,5) (Juntar [Mover (0,1) (Quadrado 5),
                          Quadrado 4,
                          Mover (4,3) (Quadrado 2)])
```

   (a) Defina a função `conta :: Imagem -> Int` que conta quantos quadrados tem uma imagem.

   (b) Defina o programa `apaga :: Imagem -> IO Imagem` que apaga um dos quadrados da imagem *à sorte*. **Sugestão**: Use a função `randomRIO :: (Int,Int) -> IO Int` para gerar o número de ordem do quadrado a remover. Para apagar um quadrado pode substitui-lo por `Juntar []`