Mini-teste 2

Laboratórios de Informática I

29 de Outubro de 2024

Exemplos do tipo de questões propostas no mini-teste 2.

Questões 1

Para cada uma das seguintes funções: indique um tipo possível para a função; documente-a em Haddock com uma breve descrição e dois exemplos de teste; defina em HUnit três testes unitários.

1. ```
   f:: ...
   f [] = []
   f (x:xs) = f (filtra1 x xs) ++ [x] ++ f (filtra2 x xs)
            where
              filtra1 x [] = []
              filtra1 x (y:ys) = if y < x then y:filtra1 x ys else filtra1 x ys
              filtra2 x [] = []
              filtra2 x (y:ys) = if y >= x then y:filtra2 x ys else filtra2 x ys
   ```

2. ```
   f:: ...
   f [] = []
   f (h:t) = insere h (f t)
     where insere (x,y) ((x1,y1):t)
              | x<=x1 =  (x,y):(x1,y1):t
              | otherwise = (x1,y1) : (insere (x,y) t)
   ```

3. ```
   f:: ...
   f [] = []
   f (h:t) = insere h (f t)
     where insere (x,y) ((x1,y1):t)
              | x+y<=x1+y1 =  (x,y):(x1,y1):t
              | otherwise = (x1,y1) : (insere (x,y) t)
   ```

4. ```
   f :: ...
   f [] l = l
   f l [] = l
   f (x:xs) (y:ys)
     | x < y = x : f xs (y:ys)
     | otherwise = y : f (x:xs) ys
   ```

Questões 2

1. Considere o seguinte tipo de dados:

```
type Posicao = (Int,Int)
```

Defina uma função `g` que recebe duas listas com a sequência de posições de duas pessoas e calcula a lista de posições por onde ambas passaram. A lista resultado não deve ter elementos repetidos. Indique o tipo de `g` , complete o comentário da função em Haddock e defina três testes unitários em HUnit.

```
{-|
...
== Exemplos:
>>> g [(1,2),(2,3),(3,4),(1,2),(2,4)] [(1,2),(0,1),(0,2),(1,3),(2,3)]
[(1,2),(2,3)]
 ...
-}
g ::  ...
g ...
```

2. Considere o seguinte tipo de dados:

```
type Posicao = (Int,Int)
```

Defina uma função `g` que recebe uma lista com a sequência de posições por onde uma pessoa passou e uma lista de posições consideradas perigosas, e calcula o número de vezes que a pessoa passou por posições perigosas. Indique o tipo de `g` , complete o comentário da função em Haddock e defina três testes unitários em HUnit.

```
{-|
...
== Exemplos:
>>> g [(1,2),(2,3),(3,4),(1,2),(2,4)] [(1,2),(0,1),(0,2),(1,3),(2,3)]
3
 ...
-}
g ::  ...
g ...
```

3. Considere os seguintes tipos de dados:

```
type Posicao = (Int,Int)   -- coordenadas (x,y)
data Movimento = N | S | E | O deriving (Show,Eq) -- norte, sul, este, oeste
```

Defina uma função `g` que recebe a posição inicial de uma pessoa, uma lista de movimentos, e uma lista de posições consideradas perigosas, e calcula quantas vezes a pessoa passou por posições perigosas. Assuma a orientação habitual dos pontos cardeais. Indique o tipo de `g` , complete o comentário da função em Haddock e defina três testes unitários em HUnit.

```
{-|
...
== Exemplos:
>>> g (1,1) [N,E,E,S,S] [(2,2),(0,1),(3,1)]
2
 ...
-}
g ::  ...
g ...
```

4. Considere os seguintes tipos de dados:

```
type Posicao = (Int,Int)   -- coordenadas (x,y)
data Movimento = N | S | E | O deriving (Show,Eq) -- norte, sul, este, oeste
```

Defina uma função que recebe a posição inicial de uma pessoa, uma lista de movimentos e uma lista de posições proibidas e calcula a posição final da pessoa depois de executar os movimentos da lista. Sempre que um movimento posicionar a pessoa numa posição proibida, esse movimento não é executado e passa ao movimento seguinte. Assuma a orientação habitual dos pontos cardeais. Indique o tipo de `g` , complete o comentário da função em Haddock e defina três testes unitários em HUnit.

```
{-|
...
== Exemplos:
>>> g (1,1) [N,E,E,S,S] [(2,2),(1,3)]
(1,0)
 ...
-}
g ::  ...
g ...
```

5. Considere o seguinte tipo de dados:

```
type Ponto= (Float, Float)
data Figura = Circulo Ponto Float  deriving (Show, Eq)  -- centro, raio
```

Defina uma função `g` que recebe uma lista de pontos, um vector de translacção a aplicar a cada ponto e uma figura, e calcula a lista de pontos resultante da translação. Devem ser removidos da lista os pontos que ficarem dentro da figura após a translacção. Indique o tipo de `g` , complete o comentário da função em Haddock e defina três testes unitários em HUnit.

```
{-|
...
== Exemplos:
>>> g [(1,2),(2,3),(3,4),(0,5)] (-1,-1) (Circulo (0.0) 2)
```

```
[(1,2),(2,3),(-1,4)]
 ...
-}
g ::  ...
g ...
```

6. Considere os seguintes tipos de dados:

```
type Ponto = (Float, Float)   -- coordenadas (x,y)
data Movimento = N | S | E | O deriving (Show,Eq) -- norte, sul, este, oeste
data Figura = Circulo Ponto Float   deriving (Show, Eq)  -- centro, raio
```

Defina uma função `g` que recebe a posição de uma pessoa, uma lista de movimentos e uma figura e calcula a posição final da pessoa depois de executar os movimentos da lista. Os movimentos devem ser executados dentro da figura dada. Se executar um movimento implica ultrapassar os limites da figura, esse movimento não deve ser execuado e deve passar aos movimentos seguintes. Indique o tipo de `g` , complete o comentário da função em Haddock e defina três testes unitários em HUnit.

```
{-|
...
== Exemplos:
>>> g (1,1) [S,S,S,S,E,E] (Circulo (0,0) 2)
(1,-1)
 ...
-}
g ::  ...
g ...
```