Ficha 3

Programação Funcional

2015/16

1. Indique como é que o interpretador de haskell avalia as expressões das alíneas que se seguem, apresentando a cadeia de redução de cada uma dessas expressões.

   (a) Considere a seguinte definição

   ```
   p :: Int -> Bool
   p 0 = True
   p 1 = False
   p x | x > 1 = p (x-2)
   ```

   Diga, justificando, qual é o valor de `p 5`.

   (b) Considere a seguinte definição

   ```
   f l = g [] l
   g l [] = l
   g l (h:t) = g (h:l) t
   ```

   Diga, justificando, qual é o valor de `f "otrec"` .

   (c) Considere a seguinte definição

   ```
   fun (x:y:t) = fun t
   fun [x] = []
   fun [] = []
   ```

   Diga, justificando, qual é o valor de `fun [1,2,3,4,5]` .

2. Defina as seguintes funções sobre listas de tuplos:

   (a) `segundos :: [(a,b)] -> [b]` que calcula a lista das segundas componentes dos pares.

   (b) `nosPrimeiros :: (Eq a) => a -> [(a,b)] -> Bool` que testa se um elemento aparece na lista como primeira componente de algum dos pares.

   (c) `minFst :: (Ord a) => [(a,b)] -> a` que calcula a menor primeira componente.
   Por exemplo, `minFst [(10,21), (3, 55), (66,3)] = 3`

   (d) `sndMinFst :: (Ord a) => [(a,b)] -> b` que calcula a segunda componente associada à menor primeira componente.
   Por exemplo, `sndMinFst [(10,21), (3, 55), (66,3)] = 55`

(e) `sumTriplos :: (Num a, Numb, Num c) => [(a,b,c)] -> (a,b,c)` soma uma lista de triplos componente a componente.
Por exemplo, `sumTriplos [(2,4,11), (3,1,-5), (10,-3,6)] = (15,2,12)`

(f) `maxTriplo :: (Ord a, Num a) => [(a,a,a)] -> a` que calcula o maximo valor da soma das componentes de cada triplo de uma lista.
Por exemplo, `maxTriplo [(10,-4,21), (3, 55,20), (-8,66,4)] = 78`

3. Defina recursivamentes as seguintes funções sobre números inteiros não negativos:

(a) `(><) :: Int -> Int -> Int` para multiplicar dois números inteiros (por somas sucessivas).

(b) `div, mod :: Int -> Int -> Int` que calculam a divisão e o resto da divisão inteiras por subtracções sucessivas.

(c) `power :: Int -> Int -> Int` que calcula a potência inteira de um número por multiplicações sucessivas.

4. Assumindo que uma hora é representada por um par de inteiros, uma viagem pode ser representada por uma sequência de etapas, onde cada etapa é representada por um par de horas (partida, chegada):

```
type Hora = (Int,Int)
type Etapa = (Hora,Hora)
type Viagem = [Etapa]
```

Por exemplo, se uma viagem for

```
[((9,30), (10,25)), ((11,20), (12,45)) , ((13,30), (14,45))]
```

significa que teve três etapas:

- a primeira começou às 9 e um quarto e terminou às 10 e 25;
- a segunda começou às 11 e 20 e terminou à uma menos um quarto;
- a terceira começou às 1 e meia e terminou às 3 menos um quarto;

Para este problema, vamos trabalhar apenas com viagens que começam e acabam no mesmo dia.

Utilizando as funções sobre horas que definiu na Ficha 1, defina as seguintes funções:

(a) Testar se uma etapa está bem construída (i.e., o tempo de chegada é superior ao de partida e as horas são válidas).

(b) Testa se uma viagem está bem construída (i.e., se para cada etapa, o tempo de chegada é superior ao de partida, e se a etapa seguinte começa depois da etapa anterior ter terminado).

(c) Calcular a hora de partida e de chegada de uma dada viagem.

(d) Dada uma viagem válida, calcular o tempo total de viagem efectiva.

(e) Calcular o tempo total de espera.

(f) Calcular o tempo total da viagem (a soma dos tempos de espera e de viagem efectiva).