Aula Teórico-prática 4

Programação Funcional

LEI 1º ano

Considere que a informação sobre um stock de uma loja está armazenada numa lista de tuplos (com o nome do produto, o seu preço unitário, e a quantidade em stock desse produto) de acordo com as seguintes declarações de tipos:

```
type Stock = [(Produto,Preco,Quantidade)]
type Produto = String
type Preco = Float
type Quantidade = Float
```

Assuma que um produto não ocorre mais do que uma vez na lista de stock.

1. Defina as seguintes funções de manipulação e consulta do stock:
   (a) `quantos::Stock->Int`, que indica quantos produtos existem em stock.
   (b) `emStock::Produto->Stock->Quantidade`, que indica a quantidade de um dado produto existente em stock.
   (c) `consulta::Produto->Stock->(Preco,Quantidade)`, que indica o preço e a quantidade de um dado produto existente em stock.
   (d) `tabPrecos::Stock->[(Produto,Preco)]`, para construir uma tabela de preços.
   (e) `valorTotal::Stock->Float`, para calcular o valor total do stock.
   (f) `inflacao::Float->Stock->Stock`, que aumenta uma dada percentagem a todos os preços.
   (g) `omaisBarato::Stock->(Produto,Preco)`, que indica o produto mais barato e o seu preço.
   (h) `maisCaros::Preco->Stock->[Produto]`, que constroi a lista dos produtos caros (i.e., acima de um dado preço).
2. Considere agora que tem a seguinte declaração de tipo para modelar uma lista de compras:

   ```
   type ListaCompras = [(Produto,Quantidade)]
   ```

   Defina as funções que se seguem:
   (a) `verifLista::ListaCompras->Stock->Bool`, que verifica se todos os pedidos podem ser satisfeitos.
   (b) `falhas::ListaCompras->Stock->ListaCompras`, que constroi a lista dos pedidos não satisfeitos.
   (c) `custoTotal::ListaCompras->Stock->Float`, que calcula o custo total da lista de compras.
   (d) `partePreco::Preco->ListaCompras->Stock->(ListaCompras,ListaCompras)`, que parte a lista de compras em duas: uma lista com os items inferiores a um dado preço, e a outra com os items superiores ou iguais a esse preço.