Cálculo de Programas

Licenciatura em Engenharia Informática

Ficha 12

1. Defina uma generalização da função lookup que devolva o resultado num *monad* MonadPlus qualquer.

   $$\mathsf{lookupM} :: (\mathsf{MonadPlus}\ m, \mathsf{Eq}\ a) \Rightarrow a \rightarrow [(a, b)] \rightarrow m\ b$$

2. Defina a função putStr usando a função sequence em vez de recursividade explícita.

3. Defina uma computação getLine :: IO String que lê uma string do teclado. Para tal, deverá ler sucessivamente caracteres do teclado, usando o getChar :: IO Char, até que um `'\n'` seja pressionado.

4. Assumindo a existência de uma computação randomBool :: IO Bool que devolve um booleano aleatório, defina a função randomInt :: Int → IO Integer que devolve um inteiro aleatório com um determinado número de bits.

5. Para contabilizar o número que participações que cada aluno teve nas aulas práticas precisamos da seguinte função.

   $$\mathsf{cadastro} :: [\mathsf{Int}] \rightarrow [(\mathsf{Int}, \mathsf{Int})]$$

   Dada uma lista com todas as participações (cada ocorrência de um número de aluno nesta lista corresponde a uma ida ao quadro), a função cadastro gera uma lista que associa cada aluno ao número de vezes que foi ao quadro. Implemente esta função usando o *monad* estado.

6. Considere uma máquina de stack muito simples para calcular expressões aritméticas. Os comandos suportados por esta máquina são os seguintes:

   $$\begin{aligned}&\mathsf{push} :: \mathsf{Int} \rightarrow \mathsf{Comando}\ ()\\ &\mathsf{pop}\ \ :: \mathsf{Comando}\ \mathsf{Int}\\ &\mathsf{add}\ \ :: \mathsf{Comando}\ ()\\ &\mathsf{mult} :: \mathsf{Comando}\ ()\end{aligned}$$

   Como esperado, push insere um elemento no topo da stack, pop retira e devolve o elemento que está no topo da stack, e add e mult retiram dois elementos do topo da stack substituindo-os, respectivamente, pela sua soma e produto. Pretende-se implementar esta máquina usando o *monad* de estado, sendo o tipo dos comandos definido da seguinte forma:

   $$\begin{aligned}&\textbf{type}\ \mathsf{Stack} = [\mathsf{Int}]\\ &\textbf{type}\ \mathsf{Comando}\ a = \mathsf{State}\ \mathsf{Stack}\ a\end{aligned}$$

   O cálculo do valor de uma expressão é feito recorrendo à sua representação na *reverse polish notation*. Por exemplo, para determinar o valor da expressão $3+(4*2)$ deveria ser executada a seguinte sequência de comandos:

   $$\begin{aligned}&> \mathsf{evalState}\ (\mathsf{push}\ 3 \gg \mathsf{push}\ 4 \gg \mathsf{push}\ 2 \gg \mathsf{mult} \gg \mathsf{add} \gg \mathsf{pop})\ [\,]\\ &11\end{aligned}$$

   Implemente os comandos acima referidos por forma a obter este comportamento.

7. Considere a seguinte implementação de um *monad* estado que também permite executar intruções de IO:

```
newtype StateIO s a = StateIO{runStateIO :: s → IO (a, s)}
instance Monad (StateIO s) where
  return x = StateIO $ λs → return (x, s)
  x >>= f  = StateIO $ λs → do (y, s') ← runStateIO x s
                               runStateIO (f y) s'
```

   (a) Defina a função evalStateIO :: StateIO s a → s → IO a que estrai o resultado deste *monad*.

   (b) Defina a função liftIO :: IO a → StateIO s a que permite executar uma instrução de IO dentro deste *monad*.