Programação Funcional

Ficha 9

Input / Output

1. A classe `Random` da biblioteca `System.Random` agrupa os tipos para os quais é possível gerar valores aleatórios. Algumas das funções declaradas nesta classe são:

   - `randomIO :: Random a => IO a` que gera um valor aleatório do tipo `a`;
   - `randomRIO :: Random a => (a,a) -> IO a` que gera um valor aleatório do tipo `a` dentro de uma determinada gama de valores.

   Usando estas funções implemente os seguintes programas:

   (a) `bingo :: IO ()` que sorteia os números para o jogo do bingo. Sempre que uma tecla é pressionada é apresentado um número aleatório entre 1 e 90. Obviamente, não podem ser apresentados números repetidos e o programa termina depois de gerados os 90 números diferentes.

   (b) `mastermind :: IO ()` que implementa uma variante do jogo de descodificação de padrões *Mastermind*. O programa deve começar por gerar uma sequência secreta de 4 dígitos aleatórios que o jogador vai tentar descodificar. Sempre que o jogador introduz uma sequência de 4 dígitos, o programa responde com o número de dígitos com o valor correcto na posição correcta e com o número de dígitos com o valor correcto na posição errada. O jogo termina quando o jogador acertar na sequência de dígitos secreta.

2. Uma aposta do EuroMilhões corresponde à escolha de 5 *Números* e 2 *Estrelas*. Os *Números* são inteiros entre 1 e 50. As *Estrelas* são inteiros entre 1 e 9. Para modelar uma aposta destas definiu-se o seguinte tipo de dados:

   ```
   data Aposta = Ap [Int] (Int,Int)
   ```

   (a) Defina a função `valida :: Aposta -> Bool` que testa se uma dada aposta é válida (i.e. tem os 5 números e 2 estrelas, dentro dos valores aceites e não tem repetições).

   (b) Defina a função `comuns :: Aposta -> Aposta -> (Int,Int)` que dada uma aposta e uma chave, calcula quantos *números* e quantas *estrelas* existem em comum nas duas apostas

   (c) Use a função da alínea anterior para:

      i. Definir `Aposta` como instância da classe `Eq`.

      ii. Definir a função `premio :: Aposta -> Aposta -> Maybe Int` que dada uma aposta e a chave do concurso, indica qual o prémio que a aposta tem.
      Os prémios do EuroMilhões são:

Números	*Estrelas*	**Prémio**	*Números*	*Estrelas*	**Prémio**
5	2	**1**	3	2	**7**
5	1	**2**	2	2	**8**
5	0	**3**	3	1	**9**
4	2	**4**	3	0	**10**
4	1	**5**	1	2	**11**
4	0	**6**	2	1	**12**
			2	0	**13**

(d) Para permitir que um apostador possa jogar de forma interactiva:

   i. Defina a função `leAposta :: IO Aposta` que lê do teclado uma aposta. Esta função deve garantir que a aposta produzida é válida.

   ii. Defina a função `joga :: Aposta -> IO ()` que recebe a chave do concurso, lê uma aposta do teclado e imprime o prémio no ecrã.

(e) Defina a função `geraChave :: IO Aposta`, que gera uma chave válida de forma aleatória.

(f) Pretende-se agora que o programa `main` permita jogar várias vezes e dê a possiblidade de simular um novo concurso (gerando uma nova chave). Complete o programa definindo a função `ciclo :: Aposta -> IO ()`.

```
main :: IO ()
main = do ch <- geraChave
          ciclo ch

menu :: IO String
menu = do { putStrLn menutxt
          ; putStr "Opcao: "
          ; c <- getLine
          ; return c
          }
    where menutxt = unlines ["",
                             "Apostar ........... 1",
                             "Gerar nova chave .. 2",
                             "",
                             "Sair .............. 0"]
```