Aula Teórico-prática 7

Programação Funcional

LEI 1º ano

A representação habitual de números (sequência de dígitos) corresponde aos coeficientes de um polinómio (sobre a base usada). Assim o número 300245 representa o número

$$3.10^5 + 2.10^2 + 4.10 + 5$$

que não é nada mais do que o valor do polinómio

$$3.x^5 + 0.x^4 + 0.x^3 + 2.x^2 + 4.x + 5$$

no ponto 10. Note que os coeficientes são da gama $[0..b[$ em que b representa a base (no nosso caso base decimal).
Para escrever um número n numa qualquer base b, devemos encontrar os dígitos (d_0, d_1, ..., d_k) dessa base (i.e., compreendidos entre 0 e $b-1$) tais que

$$n = d_0 + d_1.b + d_2.b^2 + d_3.b^3 + \cdots + d_k.b^k$$

Note que este somatório pode ser dividido em duas partes

$$\begin{aligned} n &= d_0 + d_1.b + d_2.b^2 + d_3.b^3 + \cdots + d_k.b^k \\ &= d_0 + b.(d_1 + d_2.b + d_3.b^2 + \cdots + d_k.b^{(k-1)}) \end{aligned}$$

Como a segunda parte do somatório é divisível por b, ficamos a saber o valor do dígito d_0: é o resto da divisão inteira de n por b. Note que se `divMod n b = (x,y)` então é porque $n = y + b.x$. Destes dois factos podemos concluir que

$$\begin{aligned} y + b.x &= y + b.(d_1 + d_2.b + d_3.b^2 + \cdots + d_k.b^{(k-1)}) \\ x &= d_1 + d_2.b + d_3.b^2 + \cdots + d_k.b^{(k-1)} \end{aligned}$$

E por isso os restantes dígitos podem ser calculados aplicando o mesmo processo agora ao resultado x de dividir o número original n por b.

1. Defina uma função `intToBase :: Int -> Int -> [Int]` que, dado um número e uma base, calcula a sequência de dígitos que representam esse número nessa base.

2. Defina a função inversa `baseToInt :: [Int] -> Int -> Int` que, dada a sequência de dígitos numa dada base (e essa base) calcula o número representado. Assuma que na sequência de dígitos apresentada estes aparecem do menos significativo (d_0 na introdução acima) para o mais significativo.

3. Relembre agora o algoritmo de adição de números que aprendeu (ou devia ter aprendido) na escola primária.
   A tabela ao lado exemplifica como se somam os números 12345 e 7891. Na primeira linha representámos os *"e vai um"* ou *transporte*. Note que cada digito do resultado é obtido a partir dos correspondentes dígitos dos operandos e do transporte do dígito anterior.

0	1	1	1	0	0
	1	2	3	4	5
	+	7	8	9	1
	2	**0**	**2**	**3**	**6**

(a) Defina uma função `somaDigitos :: Int -> Int -> Int -> Int -> (Int,Int)` que, dada uma base, dois dígitos dessa base e o transporte anterior, dá como resultado o dígito respectivo e o transporte para a próxima operação.

(b) Use a função anterior para definir a função `somaBase :: Int -> [Int] -> [Int] -> [Int]` que, dada uma base e as sequências de dígitos que representam dois números nessa base, implementa o algoritmo acima para obter a sequência de dígitos que representa a soma desses números.

4. Para implementarmos o algoritmo de multiplicação, vamos primeiro resolver um problema mais simples que consiste em multiplicar dois números em que um deles é composto por um único dígito.

   Na tabela ao lado relembramos como tal pode ser feito para a multiplicação de 23456 por 7. Note que esta operação pode ser caracterizada pela adição de dois números, cujos dígitos se obtêm da conhecida (e muitas vezes menosprezada) tabuada!

	2	3	4	5	6
				×	7
1	2	2	3	4	0
	4	1	8	5	2
1	**6**	**4**	**1**	**9**	**2**

   (a) Defina a função `tabuada :: Int -> Int -> Int -> (Int,Int)` que, dada uma base e dois dígitos calcula o dígito e o transporte associados à multiplicação desses dois dígitos.

   (b) Usando a função anterior, e a soma de sequências de dígitos acima, defina a função que multiplica uma sequência de dígitos por um dígito.

   (c) Para definir a multiplicação entre duas sequências de dígitos, vamo-nos basear nos seguintes factos:

   - A multiplicação de um número pela base é uma operação muito simples: basta acrescentar o dígito 0 como menos significativo.
   - A multiplicação pode ser feita à base da multiplicação por um dígito:

$$\begin{aligned} & (a_0 + a1.b + a_2.b^2 + \cdots + a_i.b^i).(c_0 + c_1.b + c_2.b^2 + \cdots c_j.b^j) \\ = \; & (a_0 + a1.b + a_2.b^2 + \cdots + a_i.b^i).c_0 \\ & +b.((a_0 + a1.b + a_2.b^2 + \cdots + a_i.b^i).(c_1 + c_2.b + \cdots c_j.b^{j-1})) \end{aligned}$$

   Usando as funções anteriores defina então a função `multBase :: Int -> [Int] -> [Int] -> [Int]` de multiplicação de duas sequências de dígitos (numa dada base).