Funções de ordem superior

3.1 Mostre como a lista em compreensão $[f\ x \mid x \leftarrow xs,\ p\ x]$ se pode escrever como combinação das funções de ordem superior *map* e *filter*.

3.2 Usando *foldl*, defina uma função $dec2int :: [Int] \to Int$ que converte uma lista de dígitos decimais num inteiro. Exemplo: $dec2int\ [2,3,4,5] = 2345$.

3.3 A função $zipWith :: (a \to b \to c) \to [a] \to [b] \to [c]$ do prelúdio-padrão é uma variante de *zip* cujo primeiro argumento é uma função usada para combinar cada par de elementos. Podemos definir *zipWith* usando uma lista em compreensão:

$$zipWith\ f\ xs\ ys = [f\ x\ y \mid (x,y) \leftarrow zip\ xs\ ys]$$

Escreva uma definição recursiva de *zipWith*.

3.4 Mostre que pode definir função $isort :: Ord\ a \Rightarrow [a] \to [a]$ para ordenar uma lista pelo método de inserção (ver a Folha 3) usando *foldr* e *insert*.

3.5 As funções *foldl1* e *foldr1* do prelúdio-padrão são variantes de *foldl* e *foldr* que só estão definidas para listas com pelo menos um elemento (i.e. não-vazias). *Foldl1* e *foldr1* têm apenas dois argumentos (uma operação de agregação e uma lista) e o seu resultado é dado pelas equações seguintes.

$$foldl1\ (\oplus)\ [x_1,\ldots,x_n] = (\ldots(x_1 \oplus x_2)\ldots) \oplus x_n$$
$$foldr1\ (\oplus)\ [x_1,\ldots,x_n] = x_1 \oplus (\ldots(x_{n-1} \oplus x_n)\ldots)$$

(a) Mostre que pode definir as funções $maximum$, $minimum :: Ord\ a \Rightarrow [a] \to a$ do prelúdio-padrão (que calculam, respectivamente, o maior e o menor elemento duma lista não-vazia) usando *foldl1* e *foldr1*.

(b) Mostre que pode definir *foldl1* e *foldr1* usando *foldl* e *foldr*. Sugestão: utilize as funções *head*, *tail*, *last* e *init*.

3.6 A função de ordem superior $until :: (a \to Bool) \to (a \to a) \to a \to a$ está definida no prelúdio-padrão; *until p f* é a função que repete sucessivamente a aplicação de f ao argumento até que que p seja verdade. Usando *until*, escreva uma definição não recursiva da função

$$mdc\ a\ b = \mathsf{if}\ b == 0\ \mathsf{then}\ a\ \mathsf{else}\ mdc\ b\ (a `mod` b)$$

que calcula o máximo divisor comum pelo algoritmo de Euclides.

3.7 Sem consultar a especificação do Haskell 98, escreva definições não-recursivas das seguintes funções do prelúdio-padrão:

(a) $(+\!\!+) :: [a] \to [a] \to [a]$, usando *foldr*;

(b) *concat* :: $[[a]] \to [a]$, usando *foldr*;

(c) *reverse* :: $[a] \to [a]$, usando *foldr*;

(d) *reverse* :: $[a] \to [a]$, usando *foldl*;

(e) *elem* :: $Eq\ a \Rightarrow a \to [a] \to Bool$, usando *any*.

3.8 Pretende-se que resolva este exercício sem usar *words* e *unwords* do prelúdio-padrão (pois *words* = *palavras* e *unwords* = *despalavras*).

(a) Escreva uma definição da função *palavras* :: *String* $\to$ [*String*] que decompõe uma linha de texto em palavras delimitadas por um ou mais espaços. Exemplo: *palavras* "Abra- ca- drabra!" = ["Abra-","ca-","dabra!"].

(b) Escreva uma definição da função *despalavras* :: [*String*] $\to$ *String* que concatena uma lista de palavras juntando um espaço entre cada uma. Note que *despalavras* não é a função inversa de *palavras*; encontre um contra-exemplo que justifique esta afirmação.

3.9 A função do prelúdio *scanl* é uma variante do *foldl* que produz a lista com os valores acumulados:

$$scanl\ f\ z\ [x_1, x_2, \ldots] = [z, f\ z\ x_1, f\ (f\ z\ x_1)\ x_2, \ldots]$$

Por exemplo:

$$scanl\ (+)\ 0\ [1,2,3] = [0, 0+1, 0+1+2, 0+1+2+3] = [0,1,3,6]$$

Em particular, para listas finitas xs temos que $last\ (scanl\ f\ z\ xs) = foldl\ f\ z\ xs$.

Escreva uma definição recursiva de *scanl*; deve usar outro nome para evitar colidir com a definição do prelúdio.