Primeira lista de exercícios
Introdução a programação em Haskell
Disciplina: Linguagens de Programação

Prof.: Carlos Camarão

30 de Agosto de 2018

1. Escreva função *map* `::` `(a->b) -> [a] -> [b]` que recebe função f e lista x e retorna a lista resultante de aplicar f a cada elemento de x.

2. Escreva função *mudaPraMinusc* `::` *String* `->` *String* que transforma toda letra minúscula no argumento em letra maiúscula, conservando sem mudança os demais caracteres.

   Use *map* e *toLower* (definida em *Data.Char*).

3. Escreva a função *dropWhile* `::` `(a->Bool) -> [a] -> [a]`, que recebe um predicado (função que retorna valor de tipo *Bool*) p e uma lista x e remove cada elemento a de x para o qual $p\ a$ é verdadeiro, a partir do primeiro, até encontrar um valor b para o qual $p\ b$ é falso.

   Exemplos: *dropWhile* `(<0) [-1,2,-3,4]` retorna `[2,-3,4]`.

   dropWhile `(>0) [-1,2,-3,4]` retorna `[-1,2,-3,4]`.

4. Escreva a função *break* `::(a -> Bool) -> [a] -> ([a],[a])`, que recebe um predicado p e uma lista x e particiona a lista x em duas sublistas *x1*, *x2*, sendo *x1* a lista contendo os elementos de x até que o predicado seja verdadeiro e *x2* o restante dos elementos de x, do elemento para o qual o predicado retorna verdadeiro seguido dos elementos restantes de x.

   Exemplos: *break* `(<0) [-1,2,-3,4]` retorna `([],[-1,2,-3,4])`.

   break `(>0) [-1,2,-3,4]` retorna `([-1],[2,-3,4])`.

5. Considere a função *splitWith* `::` `(a->Bool) -> [a] -> [[a]]`, definida a seguir:

```
splitWith p = splitWith' p . dropWhile p

splitWith' p []    = []
splitWith' p x     = x1 : splitWith p x2
  where (x1,x2) = break p x
```

Defina a função *words* `::` *String* `->` `[`*String*`]`, que recebe uma cadeia de caracteres s e retorna a lista de palavras em s.

Uma palavra é separada de outra por um ou mais caracteres c para os quais *isSpace* c retorna verdadeiro (*True*).

A função *isSpace* é definida em *Data*`.`*Char*.

6. Escreva funcão *contaOcs* `::` `[`*String*`]` `->` `[(`*Int*`,`*String*`)]` que recebe uma lista ordenada de cadeias de caracteres xs e retorna uma lista de pares (n, x) onde n é o número de vezes que x ocorre na lista ordenada ($n > 0$).

   Exemplo: *contaOcs* `["a", "a", "b", "c", "c", "c", "d"]` retorna:

   ```
   [(2,"a"), (1,"b"), (3,"c"), (1,"d")]
   ```

7. Escreva função *concat* `::` `[[`a`]]` `->` `[`a`]`, que recebe uma lista de listas x_i $(i = 1, \ldots, n)$ e retorna o resultado de concatenar os elementos em todas as listas x_i.

   Exemplo: *concat* `[[1,2],[3,4,5],[6]]` retorna `[1,2,3,4,5,6]`.

8. Escreva função *palavrasMaisComuns* `::` *Int* `->` *String*`->` *String* que recebe um inteito n e uma cadeia de caracteres *texto* e retorne a lista das n palavras mais comuns em *texto*; para cada uma das n palavras mais comuns deve ser retornda uma linha contendo a palavra e o número de ocorrências dessa palavra em *texto*.

   Use: *mudaPraMinusc*, *words*, *sort*, *contaOcs*, *sortBy*, *take*, *map*, *concat*: encadeie chamadas dessas funções com a função `(.)` de composição de funções.