Universidade Federal de Uberlândia
Faculdade de Computação
Professor: Wendel Melo

Quarta lista de exercícios de Programação Funcional (2017-1)

Questão 1: Escreva uma função em Haskell, **sem o uso direto de recursão**, que receba um número e retorne o fatorial do número recebido.

Questão 2: Escreva uma função em Haskell, **sem o uso direto de recursão**, que receba um número e retorne uma lista com todos os divisores desse número.

Questão 3: Escreva uma função em Haskell, **sem o uso direto de recursão**, que receba um número e retorne *True* se o número for primo, e *False* caso contrário.

Questão 4: Em Matemática, um número *perfeito* é um número natural para o qual a soma de todos os seus divisores positivos proprios (excluindo ele mesmo) é igual ao próprio número. Por exemplo, o número 6 é um número perfeito, pois:

$$6 = 1 + 2 + 3$$

O próximo número perfeito é o 28, pois:

$$28 = 1 + 2 + 4 + 7 + 14.$$

Quando a soma dos divisores próprios de um número n é inferior a n, dizemos que n é *deficiente*. Por outro lado, quando esta soma é maior que n, n é dito *abundante*. Escreva uma função em Haskell, **sem o uso direto de recursão**, que receba um número n e retorne uma *String* informando se o mesmo é deficiente, perfeito ou abundante.

Questão 5 (DESAFIO): Escreva uma função em Haskell, que receba um número n e retorne uma lista com os n primeiros números primos.

Questão 6: Escreva uma função em Haskell, **sem o uso direto de recursão**, que receba uma lista de números positivos e retorne o maior número presente na lista. Você não pode utilizar a função *maximum*. Sugestão: trabalhe com a ideia de função de acumulação.

Questão 7: Como generalizar a função do exercício anterior para que receba um lista com números positivos e negativos, e, ainda, assim, a mesma seja capaz de retornar o maior número presente na lista sem fazer uso direto de recursão.