1 Programação Funcional em Haskell — Lista 1

1. Defina a função `double :: Int -> Int`, que dado um inteiro, retorna 2 vezes o valor do argumento.

2. Defina a função `double2 :: Int -> Int`, usando `double`, que retorna 4 vezes o valor do argumento.

3. Defina a função `sel :: Bool -> Int -> Int -> Int`, que se comporta do seguinte modo: Se o valor do primeiro argumento for `false`, a função deve retornar o segundo argumento, se o valor do primeiro argumento for `true`, a função deve retorna o terceiro argumento.

4. Defina a função `max2 :: Int -> Int -> Int` que retorna o maior entre o dois parâmetros:

   - Usando a função `sel` do exercício 3.
   - Sem usar a função `sel` do exercício 3.

5. Defina `max3 :: Int -> Int -> Int -> Int` que retorna o maior entre 3 parâmetros, usando `max2`.

6. Defina a função `eq2 :: Int -> Int -> Bool` que retorna `true` se ambos os argumentos são iguais e `false` caso contrário. **OBS:** Lembre-se que Haskell **não** unifica variáveis!

7. Defina a função `diferent3 :: Int -> Int -> Int -> Bool` que retorna `true` se os 3 argumentos forem diferentes, e `false` caso contrário.

8. O algoritmo de Euclides pode ser usado para achar o maior divisor comum entre dois números. Matematicamente o algoritmo de Euclides pode ser definido do seguinte modo:

$$\begin{aligned} gcd(a, 0) &= a \\ gcd(a, b) &= gcd(b, a \ mod \ b) \end{aligned}$$

   Onde *mod* é a operação que retorna o resto da divisão.
   OBS: Em Hsakell, a função `mod` retorna o resto da divisão do primeiro argumento pelo segundo.

   (a) Implemente o algoritmo de Euclides em Haskell.

   (b) Implemente a função que computa o mínimo múltiplo comum entre dois números, definido como:

$$lcm(a, b) = \frac{a \cdot b}{gcd(a, b)}$$

9. A função 91 de McCarthy é definida do seguinte modo:

$$m(x) = \begin{cases} n - 10 & \text{se } n > 100 \\ m(m(n + 11)) & \text{se } n \leq 100 \end{cases}$$

   Defina esta função em Haskell. Qual valor esta função retorna para valores de entrada positivos menores que 101 ?

10. Uma fração pode ser representada por pares de valores inteiros, onde o primeiro elemento do par representa o numerador e o segundo elemento do par representa o denominador. Por exemplo as seguintes frações

$$\frac{3}{5}, \quad \frac{2}{8} \quad e \quad \frac{3}{4}$$

poderiam se expressas em Haskell como :

(3,5), (2,8) e (3,4)

Considerado esta representação defina:

(a) A função `multf :: (Int,Int) -> (Int,Int) -> (Int,Int)` que multiplica duas frações.
Ex.:
GHCi> multf (3,5) (2,8)
(6,40)

(b) A função `somaf :: (Int,Int) -> (Int,Int) -> (Int,Int)` que soma duas frações.
Ex.:
GHCi> somaf (1,2) (1,4)
(6,8)

(c) A função `subf :: (Int,Int) -> (Int,Int) -> (Int,Int)` que subtrai duas frações.

(d) A função `divf :: (Int,Int) -> (Int,Int) -> (Int,Int)` que divide duas frações.

(e) A função `toReal :: (Int,Int) -> Float` que converte a fração em um número real.

11. Defina a função `intercala :: [Int] -> [Int] -> [Int]` que intercala duas listas de inteiros, do seguinte modo :
GHCi>`intercala [1,2,3] [7,8]`
`[1,7,2,8,3]`

12. Defina a função `quads :: Int -> Int -> [Int]` que retorna a lista formada pelos quadrados dos números presentes no intervalo númerico definido pelo primeiro e segundo argumento. Isto é `quads 1 5` ▷ `[1,4,9,16,25]`

13. Defina a função `sumQuads :: Int -> Int -> Int` que retorna a soma dos quadrados dos números definidos pelo intervalo do primeiro e segundo parâmetro.

14. Defina uma função `divisors :: Int -> [Int]` de maneira que `divisors n` retorne todos os divisores do número `n`.

15. Um número é dito ser perfeito se ele é igual a soma de seus divisores. Defina uma função `perfect :: Int -> Bool` que retorna verdadeiro se o número passado como argumento é um número perfeito.

16. Um número é dito ser primo se ele ́divisível por `1` e por ele próprio. Defina uma função `prime :: Int -> Bool` que retorne verdadeiro se um número é primo.

17. Defina a função `primes :: Int -> [Int]` que retorna todos os primos no intervalo $[1, n]$, onde n é o parâmetro para a função `primes`.

18. Defina uma função `media :: [Int] -> Float` que calcule a média dos valores de uma lista.

19. Uma lista é um palíndromo se esta for igual ao seu inverso. Defina uma função `palindrome`, tal que `palindrome xs` retorne verdadeiro se a lista `xs` é um palíndromo. Qual o tipo desta função?

20. Defina uma função `toPalindrome` que receba uma lista e a transforme em um palíndromo caso esta não seja um. Exemplos `toPalindrome [1,2,3]` ▷ `[1,2,3,3,2,1]` e `toPalindrome [1,2,1]` ▷ `[1,2,1]`. Qual o tipo desta função?

Essa lista foi elaborada em conjunto com o prof. Elton Máximo Cardoso.