Paradigmas de Programação – Lista 6

Parte A - Funções

1. Usando apenas as funções uma ou mais dentre as funções `head`, `tail`, `take`, `drop`, `length` e `null`, e os recursos e elementos presentes nas duas primeiras aulas sobre programação funcional, implemente as funções abaixo:

(a) `second :: [a] -> Maybe a`, que recebe uma lista como parâmetro e retorna o segundo elemento da lista, se existir. Ex.:

```
ghci> second [1, 2, 3, 4, 5]        -- retorna 2
ghci> second [1]                    -- retorna "Nothing"
```

(b) `body :: [a] -> Maybe [a]`, que recebe uma lista como parâmetros e retorna todos os elementos, exceto o primeiro e o último, se possível. Ex.:

```
ghci> body [1, 2, 3, 4, 5]          -- retorna [2, 3, 4]
ghci> body [1]                      -- retorna "Nothing"
```

(c) `median :: [a] -> Maybe a`, que retorna o elemento central da lista de N elementos, se N é ímpar, ou o $N/2$-ésimo elemento, se N é par, se possível. Ex.:

```
ghci> median [1, 2, 3, 4, 5, 6]     -- retorna 3
```

2. Utilizando **obrigatoriamente** recursão, e opcionalmente guardas e os recursos apresentados nas duas primeiras aulas sobre programação funcional, implemente as funções abaixo:

(a) `parity :: Int -> Int`, que recebe um inteiro positivo n e retorna 0 se n é par, ou 1, se n é ímpar.

(b) `remainder :: Int -> Int -> Int`, que recebe dois inteiros positivos a e b e retorna o resto da divisão euclidiana de a por b.

(c) `quotient :: Int -> Int -> Int`, que recebe dois inteiros positivos a e b e retorna o quociente da divisão euclidiana de a por b.

(d) `greater_common_divisor :: Int -> Int -> Int`, que recebe dois inteiros positivos a e b e retorna o maior divisor comum (a, b) de a e b. Dica: $(a, b) = (b, a - b)$ e $(a, 0) = a$.

(e) `semifactorial :: Int -> Int`, que recebe um inteiror positivo e retorna

$$\texttt{semifactorial}\ n = n(n-2)(n-4)\ldots 1$$

e $\texttt{semifactorial}\ 1 = \texttt{semifactorial}\ 0 = 1$.

(f) `power :: Int -> Int -> Int`, que recebe dois inteiros positivos a e n e retorna a^n.

(g) `sum_of_first_n :: Int -> Int`, que recebe um natural n, e retorna a soma dos primeiros n números naturais.

(h) `sum_of_first_n_squares :: Int -> Int`, que recebe um natural n, e retorna a soma dos quadrados dos primeiros n números naturais.

(i) `arithmetic_progression_sum :: Int -> Int -> Int -> Int`, que recebe três naturais a, r e n e retorna a soma dos primeiros n termos de uma progressão aritmética cujo primeiro termo é igual a a e a razão é igual a r.

(j) `geometric_progression_sum:: Int -> Int -> Int -> Int`, que recebe três naturais a, q e n e retorna a soma dos primeiros n termos de uma progressão geométrica cujo primeiro termo é igual a a e a razão é igual a q.

(k) `binomial :: Int -> Int -> Int`, que recebe dois inteiros não-negativos n e m, com $n \geq m$, e computa o coeficiente binomial $\binom{n}{m}$.

(l) `count_vowels :: String -> Int`, que recebe uma string de caracteres minúsculos s e retorna o número de vogais ("$aeiou$") presentes em s.