Definição de funções simples

1.1 Considere as seguintes definições de funções:

```
incr, triplo :: Integer -> Integer
incr x = x+1
triplo x = 3*x
boasVindas :: String -> String
boasVindas nome = "Olá, " ++ nome ++ "!"
```

Smplifique as expressões seguintes o máximo possível efectuando reduções passo-a-passo. Pode verificar os resultados comparando-os com as respostas do interpretador.

(a) `incr (triplo 3)`

(b) `triplo (incr 3)`

(c) `boasVindas "Linguagem" ++ " Haskell"`

(d) `boasVindas ("Linguagem" ++ " Haskell")`

(e) `boasVindas (boasVindas "Haskell")`

1.2 Para que três valores possam ser medidas dos lados de um triângulo deve verificar-se a seguinte condição: qualquer dos valores deve ser inferior à soma dos outros dois. Complete a definição de uma função que testa esta condição; o resultado deve ser um valor boleano (`True` ou `False`).

```
testaTriangulo :: Float -> Float -> Float -> Bool
testaTriangulo a b c = ...
```

1.3 Podemos calcular a área A de um triângulo de lados a, b, c usando a fórmula de Heron:

$$A = \sqrt{s(s-a)(s-b)(s-c)}\,,$$

onde $s = (a+b+c)/2$. Complete a seguinte definição em Haskell duma função para calcular esta área.

```
areaTriangulo :: Float -> Float -> Float -> Float
areaTriangulo a b c = ...
```

1.4 Usando as funções `length`, `take`, `drop` apresentadas na primeira aula, escreva uma função `metades` que divide uma lista em duas com metade do comprimento (aproximadamente). Exemplo:

```
metades [1,2,3,4,5,6,7,8] == ([1,2,3,4], [5,6,7,8])
```

Experimente a sua definição no interpretador e investigue o acontece se a lista tiver comprimento ímpar.

1.5 Neste exercício pretende-se que use as funções o prelúdio-padrão de processamento de lista apresentadas na primeira aula: `head`, `tail`, `length`, `take`, `drop` e `reverse`.

(a) Mostre que a função `last` (que obtém o último elemento de uma lista) pode ser escrita como composição de algumas das funções acima. Consegue encontrar *duas* definições diferentes?

(b) Analogamente, mostre que a função `init` (que remove o último elemento duma lista) pode ser definida usando as funções acima de duas formas diferentes.

1.6 Os coeficientes binomiais $\binom{n}{k}$ são os números que aparecem como coeficientes dos termos X^k na expansão de $(1+X)^n$; correspondem também ao *número de formas distintas de escolher k objetos entre n* (ou, equivalentemente, o número de subconjuntos com k elementos que podemos formar de um conjunto de n elementos). Neste exercício pretende-se calcular estes coeficientes para quaisquer n e k.[1]

(a) Complete a definição duma função

```
binom :: Integer -> Integer -> Integer
binom n k = ...
```

para calcular o coeficientes binomial de n e k pela seguinte fórmula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Sugestão: pode exprimir $n!$ como `product [1..n]`.

(b) Podemos escrever a fórmula acima para calcular o mesmo resultado mas evitando multiplicações desnecessárias. Por exemplo, podemos calcular $\binom{10}{2}$ sem ter de calcular $10!$ e $8!$:

$$\binom{10}{2} = \frac{10!}{2! \times 8!} = \frac{10 \times 9 \times 8!}{2 \times 1 \times 8!} = \frac{10 \times 9}{2} = 45$$

Usando esta simplificação escreva uma definição alternativa `binom'` com o mesmo tipo e que produz os mesmos resultados mas efetuando menos cálculos.

Sugestão: Se $k < n-k$, o numerador reduz-se a ao produto dos números de $n-k+1$ até n e o denominador a $k!$. No caso em que $k \geq n-k$, o numerador reduz-se ao produto dos números de $k+1$ até n e o denominador a $(n-k)!$.

[1] Usamos inteiros de precisão arbitrária (`Integer`) para evitar "overflow" nos produtos.