1. A composição de funções define-se, em Haskell, tal como na matemática:

$$(f \cdot g)\ x = f\ (g\ x)$$

(a) Calcule $(f \cdot g)\ x$ para os casos seguintes:

$$\left\{ \begin{array}{l} f\ x = 2 * x \\ g\ x = x + 1 \end{array} \right. \quad \left\{ \begin{array}{l} f = \mathsf{succ} \\ g\ x = 2 * x \end{array} \right. \quad \left\{ \begin{array}{l} f = \mathsf{succ} \\ g = \mathsf{length} \end{array} \right. \quad \left\{ \begin{array}{l} g\ (x, y) = x + y \\ f = \mathsf{succ} \cdot (2*) \end{array} \right.$$

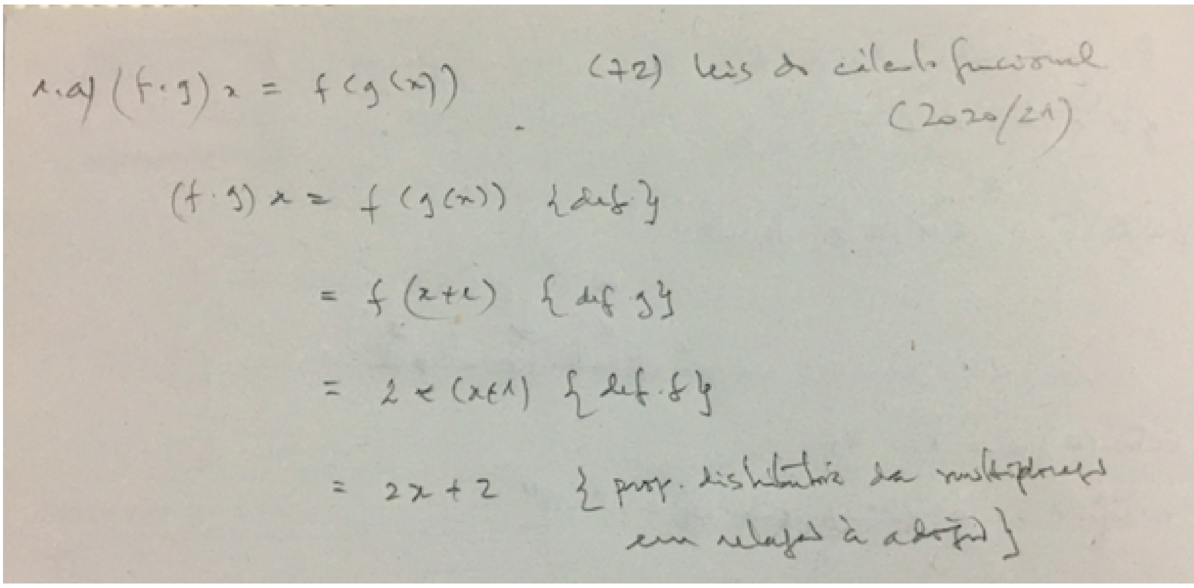
Anime as composições funcionais acima num interpretador de Haskell.

(b) Mostre que $(f \cdot g) \cdot h = f \cdot (g \cdot h)$, quaisquer que sejam f, g e h.

(c) A função $id :: a \rightarrow a$ é tal que $id\ x = x$. Mostre que $f \cdot id = id \cdot f = f$ qualquer que seja f.

Resolução

$(f \cdot g)\ x = f(g\ x) = f(x+1) = 2 * (x+1) = 2x + 2$



In [1]:

```
f x = 2 * x
g x = x + 1

-- type checking

:t f
:t g
:t (f . g)
```

f :: forall a. Num a => a -> a

g :: forall a. Num a => a -> a

(f . g) :: forall c. Num c => c -> c

In [2]:

```
-- testing for x = 5

(f . g) 5 == 2 * (5 + 1)
```

True

Resolução

$(f \cdot g)\ x = f(g\ x) = f(2 * x) = succ(2 * x)$

In [3]:
```
f = succ
g x = 2 * x

-- type checking

:t f
:t g
:t (f . g)
```

f :: forall a. Enum a => a -> a

g :: forall a. Num a => a -> a

(f . g) :: forall c. (Enum c, Num c) => c -> c

In [4]:
```
-- testing for x = 5

f(g 5) == succ (2 * 5)
```

True

Resolução

$(f \cdot g)\ x = f(g\ x) = f(\text{length}(x)) = \text{succ}(\text{length}(x))$

In [5]:
```
f = succ
g = length

-- type checking

:t f
:t g
:t (f . g)
```

f :: forall a. Enum a => a -> a

g :: forall (t :: * -> *) a. Foldable t => t a -> Int

(f . g) :: forall (t :: * -> *) a. Foldable t => t a -> Int

In [6]:
```
-- testing for x = [1..5]

f (g [1..5]) == succ (length [1..5])
succ (length [1..5])
(f . g) [1..5]
```

True
6
6

Resolução

$(f \cdot g)\ (x, y) = f(g\ (x, y)) = f(x + y) = succ(2 * (x + y)) = 2 * x + 2 * y + 1$

$(f \cdot g)\ (x, y)$

{ lei (72) }

$= \ f\ (g\ (x, y))$

{ def. g }

$$= \ f\ (x+y)$$

{ def. f }

$$= (succ \cdot (2*))(x+y)$$

{ lei (72) }

$$= succ\ ((2*)\ (x+y))$$

{ prop. distributiva da multiplicação em relação à adição }

$$= succ\ (2x+2y)$$

{def. $succ$}

$$= 2x+2y+1$$

In [7]:

```haskell
f = succ . (2*)
g (x,y) = x+y

-- type checking

:t f
:t g
:t (f . g)
```

f :: forall c. (Enum c, Num c) => c -> c

g :: forall a. Num a => (a, a) -> a

(f . g) :: forall c. (Enum c, Num c) => (c, c) -> c

In [8]:

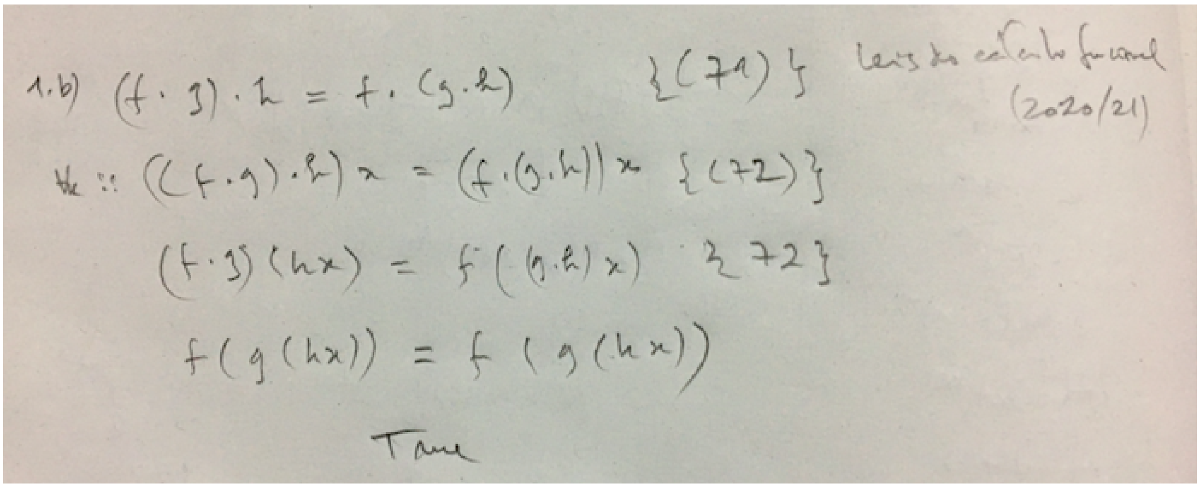
```haskell
-- testing for (x,y) = (2,3)

(f . g) (2,3) == 2*2 + 2*3 + 1
```

```
True
```

Resolução

$$(f \cdot g) \cdot h = f \cdot (g \cdot h)$$



Resolução

$f \cdot id = id \cdot f = f$

1.c) $f \cdot id = id \cdot f = f$

$$\forall x :: (f \cdot id)\ x = (id \cdot f)\ x \qquad \{(71)\}$$

$$f\ (id\ x) = id\ (f\ x) \qquad \{(72)\}$$

$$f\ x = f\ x \qquad \{(73)\}$$

True

In [9]:
```
:t id
```

```
id :: forall a. a -> a
```