Exercises

1. Define the function \( \text{length} :: [a] \rightarrow \text{Int} \)

\[
\text{length} :: [a] \rightarrow \text{Int} \\
\text{length} [] = 0 \\
\text{length} (_:xs) = 1 + \text{length} \ x
\]

2. Evaluate the expressions that don’t contain an error

\[
\text{sum} \ x \ + \ \text{length} \ x \ = \ 9 \\
\text{xs} \ +\!\! + \ \text{length} \ x \ = \ \times \\
\text{xs} \ +\!\! + \ \text{[length} \ x] \ = \ [1,2,3,3] \\
[\text{sum} \ x, \ \text{length} \ x] \ = \ [6,3] \\
[\text{xs}, \ \text{length} \ x] \ = \ \times
\]

\[
\text{xs} = [1,2,3] \\
5:\text{xs} = [5,1,2,3] \\
\text{xs}:5 = \times \\
[\text{tail} \ x,5] = \times \\
[\text{tail} \ x,\text{[5]}] = [[2,3],[5]] \\
\text{tail} \ [\text{xs},\text{xs}] = [[1,2,3]]
\]
3. Is the function \( \text{th} \) well defined? If so, what does it do and what is its type?

\[
\text{th} :: [[a]] \rightarrow [a]
\]

\[
\text{th} = \text{tail} . \text{head}
\]

\[
(\cdot) :: (b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow c
\]

4. What does the expression \( \text{map f} . \text{map g} \) compute? How can it be rewritten?

\[
\text{map f} . \text{map g} = \text{map (f . g)}
\]
Exercises

5. Implement `revmap` using pattern matching

```haskell
revmap :: (a → b) → [a] → [b]
revmap f [] = []
revmap f (x:xs) = revmap f xs ++ [f x]
```

6. Implement `revmap` using function composition

```haskell
revmap :: (a → b) → [a] → [b]
revmap f = reverse . map f
```
Exercises

7. Find expressions to ...

- increment elements in `xs` by 1
- increment elements in `ys` by 1
- find the last element in `xs`

```
xs = [1,2,3]
ys = [xs,[7]]
```

```
map succ xs       = [2,3,4]
map (map succ) ys = [[2,3,4],[8]]
head (reverse xs) = 3
```

8. Define the function

```
last :: [a] → a
```

```
last :: [a] → a
last [x]   = x
last (_:xs) = last xs
```

9. Evaluate all the expressions that don’t contain an error

```
map sum xs       = [6,7]
map sum ys       = [6,7]
last ys          = [7]
map last ys      = [3,7]
last (last ys)   = 7
```