### Exercises

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

   ```haskell
   \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**

   - \( \text{sum xs} + \text{length xs} \; = \; 9 \)
   - \( \text{xs ++ \; length \; xs} \; \times \)
   - \( \text{xs ++ \; [length \; xs]} \; = \; [1,2,3,3] \)
   - \( \text{[sum \; xs, \; length \; xs]} \; = \; [6,3] \)
   - \( \text{[xs, \; length \; xs]} \; \times \)

   ```haskell
   \text{sum} \; :: \; \text{[Int]} \rightarrow \text{Int}
   \text{sum} \; [\;] \; = \; 0
   \text{sum} \; (x:x:s) \; = \; x \; + \; \text{sum} \; x
   ```

   ```haskell
   \text{length} \; = \; \text{foldr} \; (\_ \; c->c+1) \; 0
   ```

   ```haskell
   \text{sum} \; = \; \text{foldr} \; (+) \; 0
   ```

   ```haskell
   \text{xs} \; = \; [1,2,3]
   ```

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

```
th :: [[a]] → [a]
```

```
th = tail . head
```

```
(·) :: (b → c) → (a → b) → a → c
```

4. What does the expression `map f . map g` compute? How can it be rewritten?

```
map f . map g = map (f ∘ g)
```

```
head :: [a] → a
head (x:_ ) = x
```

```
tail :: [a] → [a]
tail (_:xs) = xs
```
5. Implement \texttt{revmap} using pattern matching

\begin{verbatim}
revmap :: (a -> b) -> [a] -> [b]
revmap f [] = []
revmap f (x:xs) = revmap f xs ++ [f x]
\end{verbatim}

6. Implement \texttt{revmap} using function composition

\begin{verbatim}
revmap :: (a -> b) -> [a] -> [b]
revmap f = reverse . map f
\end{verbatim}
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
map sum ys = [6, 7]
last ys = [7]
map last ys = [3, 7]
last (last ys) = 7