1. Define the function \texttt{length :: [a] \rightarrow Int}

\begin{verbatim}
length :: [a] \rightarrow Int
length [] = 0
length (_:xs) = 1 + length xs
\end{verbatim}

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

\begin{verbatim}
sum xs + length xs = 9
xs ++ length xs \times
xs ++ [length xs] = [1,2,3,3]
[sum xs, length xs] = [6,3]
[xs, length xs] \times
\end{verbatim}

\begin{verbatim}
sum :: [Int] \rightarrow Int
sum [] = 0
sum (x:xs) = x + sum xs
\end{verbatim}

\begin{verbatim}
sum = foldr (+) 0
length = foldr (\_ c->c+1) 0
\end{verbatim}

\begin{verbatim}
xs = [1,2,3]
\end{verbatim}

\begin{verbatim}
5:xs = [5,1,2,3]
xs:5 \times
[tail xs,5] \times
[tail xs,[5]] = [[2,3],[5]]
tail [xs,xs] = [[1,2,3]]
\end{verbatim}
Exercises

3. Is the function \( \text{th} \) well defined? If so, what does it do and what is its type?

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

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

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

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

\[ \text{head} (x:_ \_ ) = x \]

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

\[ \text{tail} (_ \_ :xs ) = xs \]

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

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

5. **Implement** `revmap` **using pattern matching**

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

6. **Implement** `revmap` **using function composition**

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

```
map :: (a → b) → [a] → [b]
map f []    = []
map f (x:xs) = f x : map f xs
```

```
reverse :: [a] → [a]
reverse []    = []
reverse (x:xs) = reverse xs ++ [x]
```

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

```
revmap f = map f . reverse
```
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]
y = [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