Exercises

1. *Define the function* `length :: [a] → Int`

```
length :: [a] → Int
length []     = 0
length (_:xs) = 1 + length xs
```

```
sum :: [Int] → Int
sum []     = 0
sum (x:xs) = x + sum xs
```

```
sum = foldr (+) 0
```

```
length = foldr (\_ c->c+1) 0
```

2. *Evaluate the expressions that don't contain an error*

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

```
sum xs + length xs   = 9
xs ++ length xs      ✗
xs ++ [length xs]    = [1,2,3,3]
[sum xs, length xs] = [6,3]
[xs, length xs]      ✗
```

```
5:xs              = [5,1,2,3]
xs:5              ✗
[tail xs,5]       ✗
[tail xs,[5]] = [[2,3],[5]]
tail [xs,xs]  = [[1,2,3]]
```

Exercises

3. *Is the function* `th` *well defined? If so, what does it do and what is its type?*

```
th :: ?
th = tail . head
```

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

```
th :: [[a]] → [a]
```

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

```
tail :: [a] → [a]
tail (_:xs) = xs
```

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

```
map f . map g = map (f . g)
```

Exercises

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

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

```
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]
```

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

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

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

```
revmap f = map f . reverse
```

Exercises

7. *Find expressions to …*

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

… *increment elements in xs by 1*

… *increment elements in ys by 1*

… *find the last element in xs*

```
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
```