Definitions

```
iter :: Int -> (a -> a) -> (a -> a)
iter n f
| n > 0 = f . iter (n - 1) f
| otherwise = id
```

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

```
elem :: a -> [a] -> a
elem y [] = False
elem y (x:xs) = if y == x then True else elem y xs
```

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + length xs
```

```
(++) :: [a] -> [a] -> [a]
(++) [] [] = []
(++) [] (y:ys) = y : ((++) [] ys)
(++) (x:xs) ys = x : ((++) xs ys)
```

```
map f [] = []
map f (x:xs) = f x : map f xs
```

```
abs x
| x < 0 = abs (-x)
| otherwise = x
```

```
signum x
| x < 0 = -1
| x == 0 = 0
| otherwise = 1
```

Exercise 9.5

Prove for all finite lists `xs` and `ys`: `sum (xs ++ ys) = sum xs + sum ys` where

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

```
[] ++ ys = ys
(x:xs) ++ ys = x : (xs ++ ys)
```

Exercise 9.6

Prove for all finite lists `xs`: `xs ++ [] = xs`
Prove for all finite lists `xs`: `(xs ++ ys) ++ zs = xs ++ (ys ++ zs)`

Exercise 9.7

Prove for all finite lists `xs`: `sum (reverse xs) = sum xs`
Prove for all finite lists `xs`: `length (reverse xs) = length xs`

Exercise 9.8

Prove for all finite lists `xs` and `ys`: `elem z (xs ++ ys) = elem z xs || elem z ys`

Exercise 9.10

Prove for all finite lists `xs` and defined `n`: `take n xs ++ drop n xs = xs`, where

```
take 0 _ = []
take _ [] = []
take n (x:xs) = x :  take (n-1) xs
```

```
drop 0 xs = xs
drop n [] = []
drop n (x:xs) = drop (n-1) xs
```

Exercise 11.25

Prove for all `x`: `f.(g.h) x == (f.g).h x`

Exercise 11.26

Prove for all `f`: `id.f = f`

Exercise 11.29

Prove for all natural `n`: `iter n id = id`

Exercise 11.30

Prove for all natural `x`: `abs.abs x = abs x`
Prove for all natural `x`: `signum.signum x = signum x`

Exercise 11.31

Prove: `map f (ys ++ zs) = map f ys ++ map f zs`

Exercise 11.34

Prove: `concat (map (map f) xs) = map f (concat xs)`

Exercise 11.35

Prove: `(0<)  .  (+1) = (0<=)`