Aufgabe 8-1 Funktionen höherer Ordnung

Implementieren Sie folgende Funktionen ohne Rekursion oder List-Comprehensions mittels Funktionen höherer Ordnung aus der Standardbibliothek wie map, filter, foldr, usw.

Tipp: Nutzen Sie zum Testen während der Implementierung die Funktionen scanr bzw. scanl, um die Zwischenschritte der Fold-Funktionen besser nachvollziehen zu können.

a) Zuerst ein alter Bekannter: Implementieren Sie die Funktion length', die die Länge einer Liste berechnet.

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

foldl :: (a -> b -> b) -> b -> [a] -> b
foldl f akk [ ] = akk
foldl f akk (x : xs) = foldl f (f x akk) xs

length' : [a] -> Int
length' xs = foldl (\x akk -> 1 + akk)
```

f; 0 ↑ Startakkumulator; xs ↑

λ ⇐ griechisches Lambda

```
length' [1,2,3] = foldr f 0 [1,2,3]
                = foldr f (f 1 0) [2,3]
                = foldr f 1 [2,3]
                = foldr f (f 2 1) [3]
                = foldr f 2 [3]
                = foldr f (f 3 2) [ ]
                = foldr f 3 [ ] = 3
```

```
lenght' = foldl (\_n -> n+1) 0
```

↑ Leerzeichen

```
(\n _ -> n+1)
```

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f akk [ ] = akk
foldr f akk (x : xs) = f x (foldr f akk xs)

length' : [a] -> Int
length' = foldr (\_n -> n+1) 0
                    f
length' [100,2,3] = f 100 (foldr f 0 [2,3])
                  = 1 + foldr f 0 [2,3]
                  =
```

miro

a) Implementieren Sie die Typklasse `Monoid` für den Typ `Ordering`.
Stellen Sie sich bei der Bearbeitung folgende Fragen:

- Welches der drei Werte von `Ordering`, `LT, EQ, GT` ist das neutrale Element und warum?
- Wie muss sich die `mappend` Funktion verhalten, um das Ergebnis der "vorangehenden" Vergleiche in den anderen Fällen zu erhalten?
  (D.h. das Ergebnis späterer Vergleiche, wie der Vergleich nach der Länge des Namens bei "Miller" und "Müller", als irrelevant zu ignorieren, da bereits der zweite Buchstabe unterschiedlich ist)
- Welche Eigenschaft(en) muss die `mappend` Funktion auf jeden Fall erfüllen, damit die Monoid-Eigenschaft erfüllt ist?

Hinweis: Die Typklasse `Monoid` ist für `Ordering` bereits standardmäßig implementiert, d.h. Sie müssen diese Teilaufgabe "freihand" ohne Evaluation durch ghc implementieren (Sie erhalten sonst eine Fehlermeldung "error: Duplicate instance declarations"). In der nächsten Teilaufgaben verwenden wir dann die Standardimplementierung.

```
data Ord = LT | EQ | GT

class Monoid m
  mempty :: m
  mappend :: m -> m -> m
  mconcat :: [m] -> m
  mconcat = foldr mappend mempty
```

$*: M \times M \to M$

$(M, *)$ ist Monoid $:\Leftrightarrow$

$*$ ist assoziativ, d.h.
$\forall a,b,c \in M: (a*b)*c = a*(b*c)$

$\exists e \in M \forall a \in M:$
$e*a = a = a*e$

$(\mathbb{N}_0, +)$ ist Monoid mit $e = 0$

$(\mathbb{Z}, \cdot)$ ist Monoid mit $e = 1$

$(\mathbb{N}_{\geq 1}, +)$ ist kein Monoid

```
data MyList a = Empty | Node a (MyList a)

instance Monoid MyList a where
  mempty = Empty
  mappend Empty xs = xs
  mappend xs Empty = xs
  mappend (Node x xs) ys = Node x (mappend xs ys)
```

x: head, mappend xs ys: tail

instance

$([1,2,3] ++ [4,5]) ++ [7,8] = [1,2,3] ++ ([4,5] ++ [7,8])$

Assoziativgesetz ist erfüllt!

b) Ab hier arbeiten wir mit dem vordefinierten Typ `Ordering`. Der oben im Text skizzierte Vergleich lässt sich ohne Verwendung der Monoid-Eigenschaft folgendermaßen implementieren:

```
revCompare :: String -> String -> Ordering
revCompare []     []     = EQ
revCompare []     (_:_)  = LT
revCompare (_:_)  []     = GT
revCompare (x:xs) (y:ys) = case flip compare x y of
                             EQ    -> revCompare xs ys
                             other -> other
```

Schreiben Sie die Funktion so um, dass Sie sich bei `revCompare (x:xs) (y:ys)` die Monoid-Eigenschaft von `Ordering` aus Aufgabe a) zunutze machen.

class Ordering a
compare :: a → a → Ord

compare :: (Ordering a) ⇒ a → a → Ordering

flip :: (a → b → c) → b → a → c
flip f x y = f y x

pow :: Int → Int → Int
pow b 0 = 1
pow b n = b * pow b (n-1)

pow 2 3 = $2^3 = 2 \cdot 2 \cdot 2 = 8$
flip pow 2 3 = $3^2 = 9$