```haskell
-- Useful functions

-- map: applies a function to every element in a list
map :: (s -> t) -> [s] -> [t]
map f [] = []
map f (x:xs) = f x : map f xs

-- filter: filters a list using a pred function
pred :: t -> Bool

filter :: (t -> Bool) -> [t] -> [t]
filter pred [] = []
filter pred (x:xs) = if pred x then x : (filter pred xs)
                     else filter pred xs
-- fold version
filter' :: (t -> Bool) -> [t] -> [t]
filter' pred = foldr check []
        where check x xs = if (pred x) then x:xs else xs

-- composition of functions, returning a function
comp :: (u -> t) -> (s -> u) -> (s -> t)
comp f g = (\x -> f (g x))

-- apply a function multiple times, return a function
iter :: (t -> t) -> Integer -> (t -> t)
iter f n
        | (n == 0) = (\x -> x)
        | otherwise = f . (iter f (n - 1))
-- == \x -> f ((iter f (n - 1)) x)

-- foldr
foldr :: (s -> t -> t) -> t -> [s] -> t
foldr op i [] = i
foldr op i (x:xs) = op x (foldr op i xs)

-- foldl
foldl :: (t -> s -> t) -> t -> [s] -> t
foldl op i [] = i
foldl op i (x:xs) = foldl op (op i x) xs

-- returning length of list
length :: [t] -> Int
length list = foldr (+) 0 (map (\x -> 1) list)

--
length' :: [t] -> Int
```

```haskell
length = foldr (\x n -> n + 1) 0

-- sentenceLength
sentenceLength :: [String] -> Int
sentenceLength :: foldr (\l n -> length l + n) 0

-- append lists
app :: [t] -> [t] -> [t]
app [] r = r
app (x:xs) r = x:(app xs r)

-- fold version
app' :: [t] -> [t] -> [t]
app' left right = foldr (:) right left

-- reverse list
rev :: [t] -> [t]
rev [] = []
rev (x:xs) = app (rev xs) [x]

-- fold version
rev' :: [t] -> [t]
rev' = foldl cons []
        where cons xs x = x:xs

-- flattens a list, reducing its depth by 1
flatten :: [[t]] -> [t]
flatten = foldr app []

-- zipWith: combines two lists with a given function
zipWith :: (s -> t -> u) -> [s] -> [t] -> [u]
zipWith f (x:xs) (y:ys) = f x y : zipWith xs ys
zipWith _ _ _ = []



-- List comprehension
triples :: [(Int, Int, Int)]
triples = [(x, y, z) | z <- [1..], y <- [1..z-1], x <- [1..z-1], x^2 + y^2 == z^2]
```