The function \texttt{map}

\texttt{suclist :: [Int] -> [Int]}
\texttt{suclist [] = []}
\texttt{suclist (x:xs) = suc x : suclist xs}

\texttt{sqrtlist :: [Float] -> [Float]}
\texttt{sqrtlist [] = []}
\texttt{sqrtlist (x:xs) = sqrt x : sqrtlist xs}

\texttt{map :: (a -> b) -> [a] -> [b]}
\texttt{map g [] = []}
\texttt{map g (x:xs) = g x : map g xs}

\begin{align*}
\texttt{suclist :: [Int] -> [Int]} & \quad \texttt{sqrtlist :: [Float] -> [Float]} \\
\texttt{suclist = map suc} & \quad \texttt{sqrtlist = map sqrt}
\end{align*}
The function filter

```haskell
dropEven : [Int] -> [Int]
dropEven [] = []
dropEven (x:xs) | odd x = x : dropEven xs
                | otherwise = dropEven xs

dropUpper : [Char] -> [Char]
dropUpper [] = []
dropUpper (x:xs) | isLower x = x : dropUpper xs
                 | otherwise = dropUpper xs

filter : (a -> Bool) -> [a] -> [a]
filter g [] = []
filter g (x:xs) | g x = x : filter g xs
                 | otherwise = filter g xs
```

\[
\downarrow
\]

\[
\begin{align*}
dropEven &: [Int] \rightarrow [Int] \\
dropEven &= \text{filter odd} \\
dropUpper &: [Char] \rightarrow [Char] \\
dropUpper &= \text{filter isLower}
\end{align*}
\]
The function `foldr`

```
sum :: Num a => [a] -> a
sum [] = 0
sum (x:xs) = x + sum xs

prod :: Num a => [a] -> a
prod [] = 1
prod (x:xs) = x * prod xs

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

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr g e [] = e
foldr g e (x:xs) = g x (foldr g e xs)
```

```
sum :: Num a => [a] -> a
sum = foldr (+) 0

prod :: Num a => [a] -> a
prod = foldr (*) 1

concat :: [[[a]]] -> [a]
concat = foldr (++) []
```