Mapping vs folding lists

MAP: apply a function to each element of a list

		function		takes initial list		returns a new list
map	::	(a -> b)	->	[a]	->	[b]
map		_		[ ]	=	[ ]
map		f		(x:xs)	=	f x : map f xs
map		(+1)		[1, 2, 3]	=	[1+1, 2+1, 3+1] = [2, 3, 4]

FOLD: reduce a list by replacing each cons with the function

		function		base case		takes initial list		returns a value
foldr	::	(a -> b -> b)	->	b	->	[a]	->	b
foldr		_		z		[ ]	=	z
foldr		f		z		(x:xs)	=	f x (foldr f z xs)
foldr		(+)		0		[1, 2, 3]	=	1+ 2+ 3+ 0 = 6