Here is a function that, given a list of strings, produces a list of their lengths:

```scheme
;; lengths : [ListOf String] -> [ListOf Number]
;; compute the lengths of all the strings in the given list
(define (lengths ls)
  (cond [(empty? ls) empty]
        [else (cons (string-length (first ls))
                   (lengths (rest ls)))]))
```

(check-expect (lengths empty) empty)
(check-expect (lengths (list "yes" "no")) (list 3 2))

Rewrite the `lengths` function to use the `map` function in its body. Remember that the signature and purpose for `map` are:

```scheme
;; map : [X -> Y] [ListOf X] -> [ListOf Y]
;; apply the given function to every element of the given list
```