Steps

1. Structural recursion template for (listof X)
2. Wrapping in local with trampoline
3. Adding accumulator parameter
   - As argument to inner function
   - After every …
   - Any call to the inner function
   - Acc. value in the trampoline as …
4. Specify the type, invariant, and examples of acc.
5. Complete the function
   - Don't forget to **Initialize, update, exploit** the accumulator

1:Structural recursion template for (listof X)

```
(define (fn-for-lox lox)
  (cond [(empty? lox) (...)]
        [else
         (... (first lox)
              (fn-for-lox (rest lox)))]))
```

2:Wrapping in local with trampoline

```
(define (outer-fn lox0)
  (local [(define (fn-for-lox lox)
            (cond [(empty? lox) (...)]
                  [else
                   (... (first lox)
                        (fn-for-lox (rest lox)))]))]
    (fn-for-lox lox0)))
```

3:Adding accumulator parameter

```
(define (outer-fn lox0)
  ;; acc is Type: <invariant>
  (local [(define (fn-for-lox lox acc)
            (cond [(empty? lox) (... acc)]
                  [else
                   (... acc
                        (first lox)
                        (fn-for-lox (rest lox) (... acc)))]))]
    (fn-for-lox lox0 ...)))
```