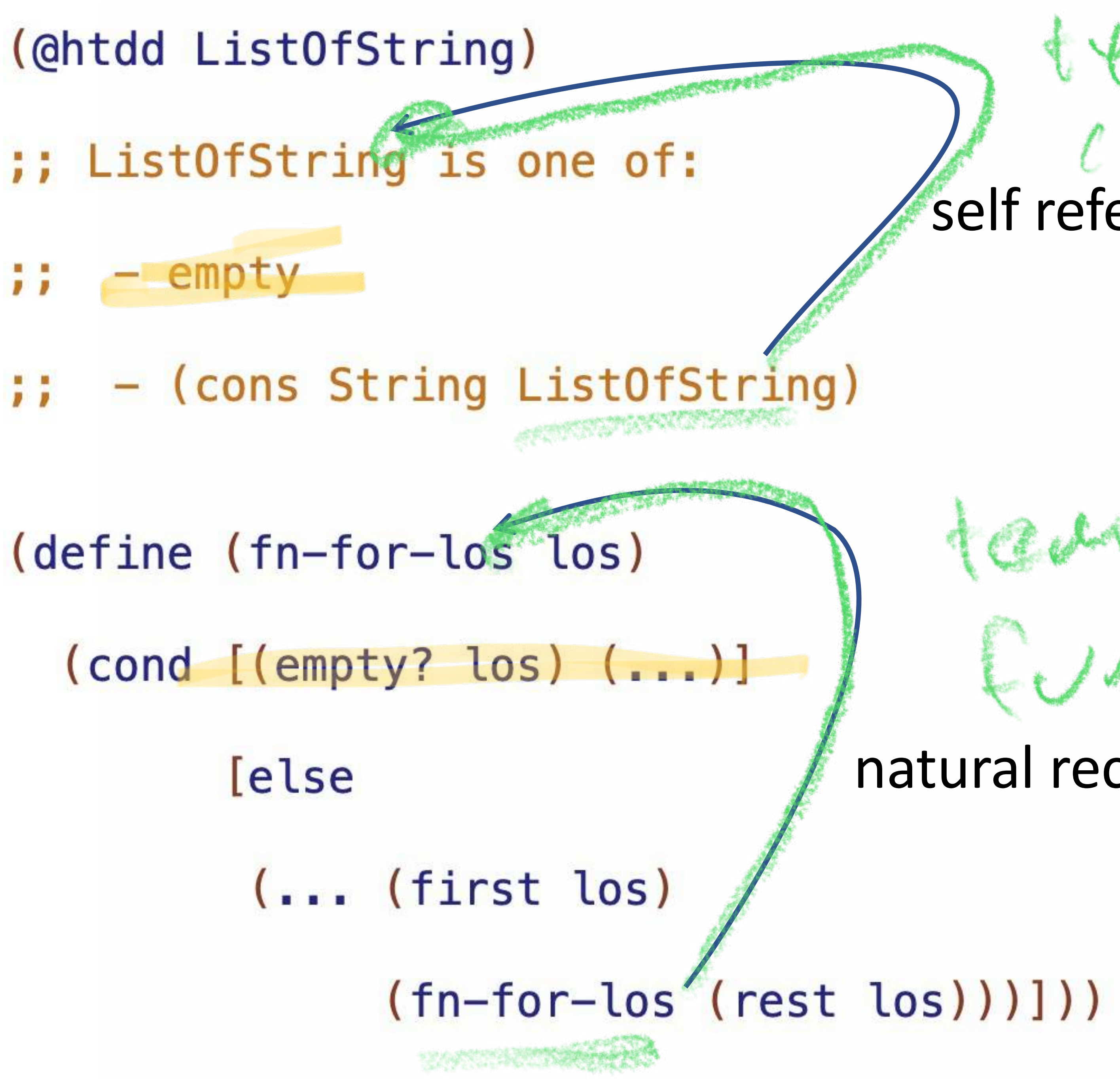
For midterm 1

- I know that the detailed instructions matter, and I will read them this week.
- I know that the midterm will be mostly autograded.
- I know that if I hand in files with “red errors” that file will receive a 0.
- I know that if I comment out @tags I will lose many points.
- I know that if I don’t follow a problem statement carefully I will lose many points.

A. I would like to get a good grade and will do all the above.

B. Nope, not gonna read the instructions, not going to pay attention to the details, my grade will be what it will be

```
(@htdd ListOfString)
;; ListOfString is one of:
;;  - empty
;;  - (cons String ListOfString)

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



```
(@htdd ListOfString)

;; ListOfString is one of:

;;  - empty

;;  - (cons String ListOfString)


(define (fn-for-los los)

  (cond [(empty? los) (...)]

        [else

         (... (first los)

              (fn-for-los (rest los)))]))
```

type comment

self reference

template function

natural recursion

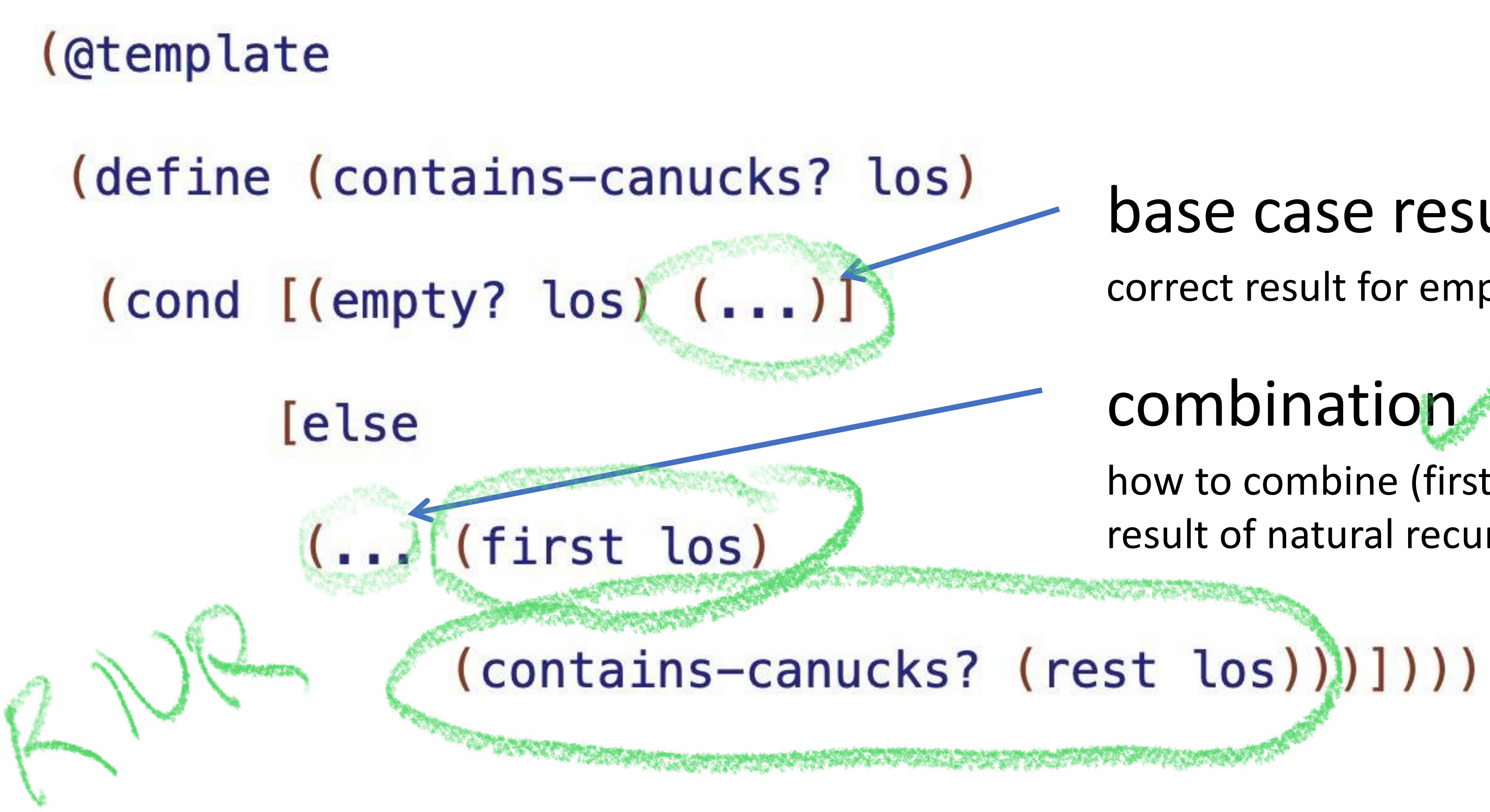
```
(@template
 (define (contains-canucks? los)
  (cond [(empty? los) (...)]
        [else
         (... (first los)
              (contains-canucks? (rest los)))])))
```

natural recursion

Trusting the Natural Recursion

result of natural recursion (RNR) will be correct if and only if

- correct base case result ✓
- correct contribution of first ✓
- correct combination of contribution and RNR ✓



```
(@template
 (define (contains-canucks? los)
  (cond [(empty? los) (...)]
        [else
         (... (first los)
              (contains-canucks? (rest los)))])))
```

base case result ✓

correct result for empty

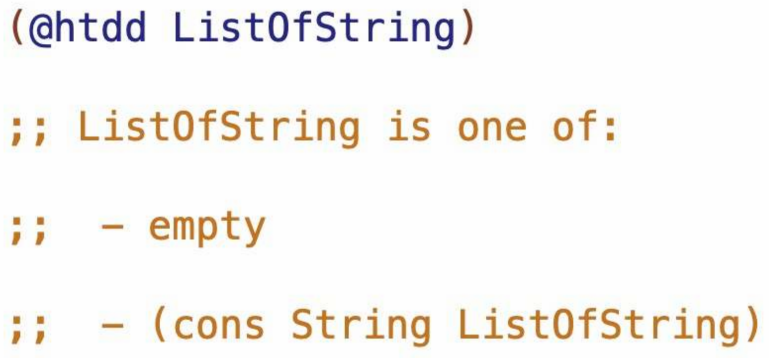
combination ✓

how to combine (first los) and result of natural recursion

HTDD

type comment

arbitrary-sized information -> requires well-formed self-referential data definition



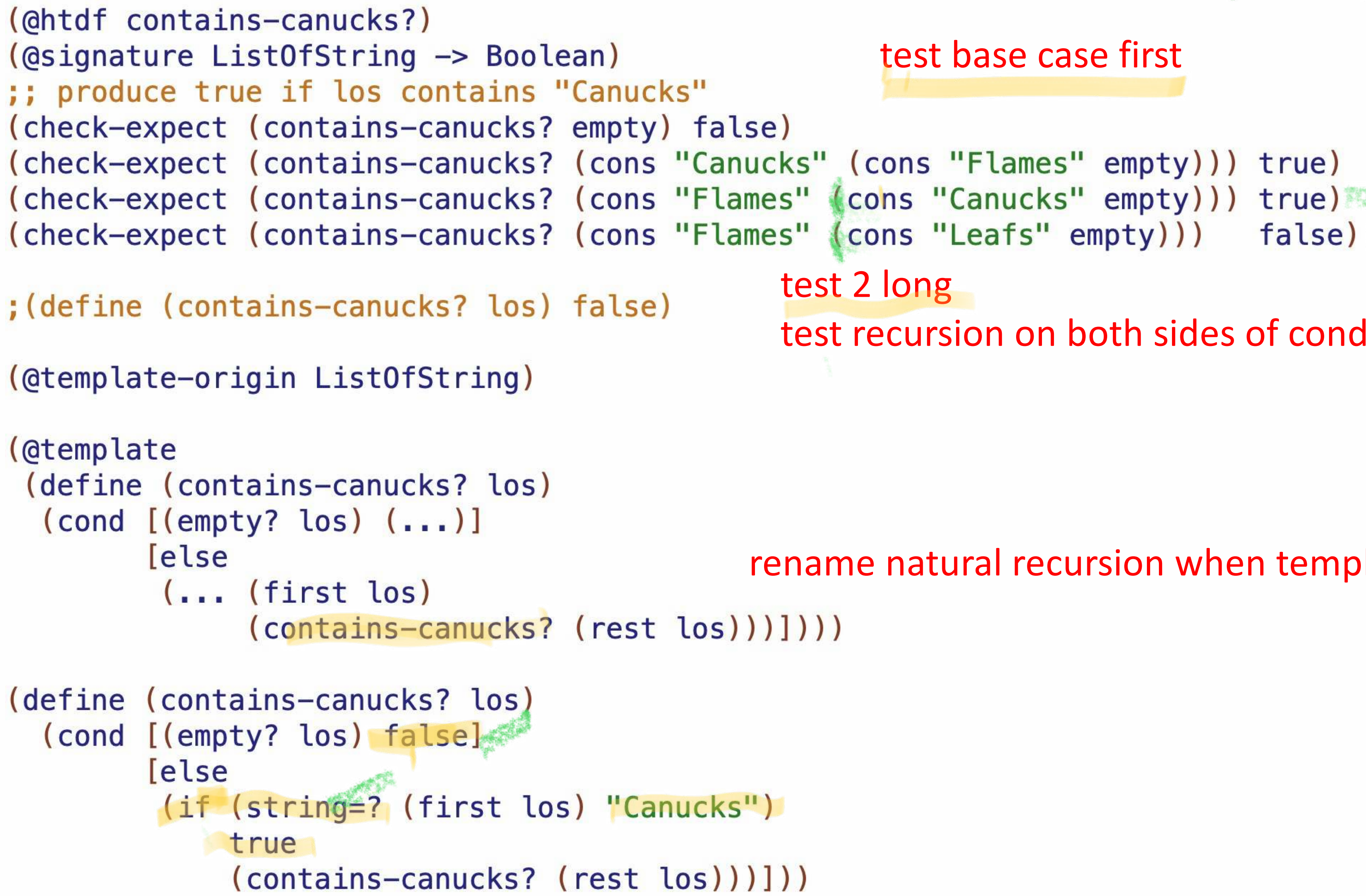
```
(@htdd ListOfString)

;; ListOfString is one of:

;;  - empty

;;  - (cons String ListOfString)
```

one of with:

one or more base subclass no SR

one or more self-reference subclasses w/ SR arbitrary-sized information -> requires self-referential data definition

H+DF



```
(@htdf contains-canucks?)
(@signature ListOfString -> Boolean)
;; produce true if los contains "Canucks"
(check-expect (contains-canucks? empty) false)
(check-expect (contains-canucks? (cons "Canucks" (cons "Flames" empty))) true)
(check-expect (contains-canucks? (cons "Flames" (cons "Canucks" empty))) true)
(check-expect (contains-canucks? (cons "Flames" (cons "Leafs" empty)))   false)
```

test base case first

test 2 long

test recursion on both sides of conditional

```
;(define (contains-canucks? los) false)

(@template-origin ListOfString)

(@template
 (define (contains-canucks? los)
  (cond [(empty? los) (...)]
        [else
         (... (first los)
              (contains-canucks? (rest los)))])))
```

rename natural recursion when templating

```
(define (contains-canucks? los)
  (cond [(empty? los) false]
        [else
         (if (string=? (first los) "Canucks")
             true
             (contains-canucks? (rest los)))]))
```

can "trust the natural recursion" if and only if:

- correct base case result
- correct contribution of first
- correc combination

Function	Base case result	Combination
sum	0	(+ <first> RNR)
product	1	(* <first> RNR)
count	0	(+ 1 RNR)
doubles	empty	( )

Function	Base case result	Combination
sum	0	(+ <first> RNR)
product	1	(* <first> RNR)
count	0	(+ 1 RNR)
doubles	empty	(cons (* 2 <first>) RNR)