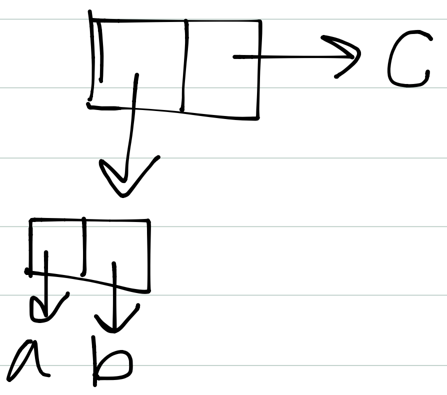
3.17

(cons (cons 'a 'b) 'c)



#pairs = 2

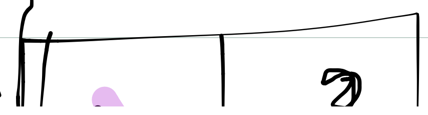
```
(+ (count (a b))  → pair
   (count (c))    → not pair
   1)
```

↓

```
(+ (count (a))
   (count (b))
   1)
```

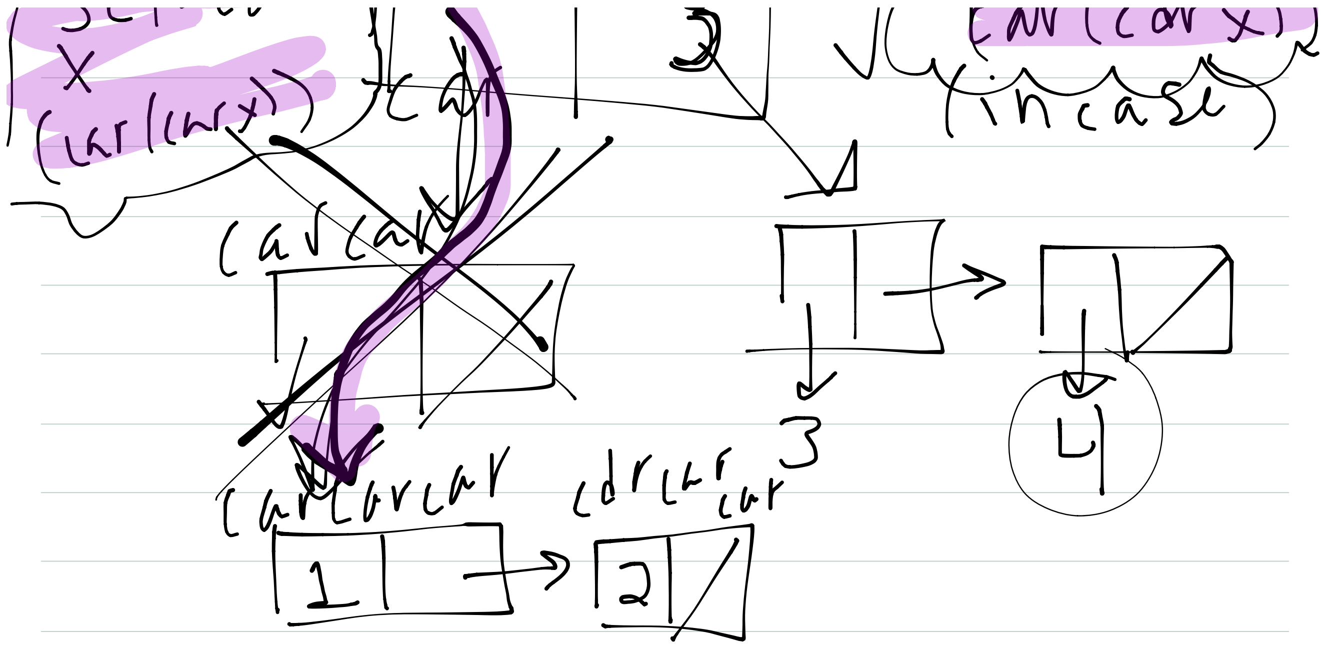
cons list '()

(cons (~~list~~ (~~cons~~ 1 2)) (list 3 4))



set-car!

set-cdr! x



~~if cdr of pair is null, don't count it as a pair~~

if the car/cdr is another pair, make this pair's car/cdr (same) ~~to be~~ the car of that pair it points to

```
(define (count-pairs x)
  (cond ~~if~~ ((not (pair? x)))
     0)
```

((begin (append! x [illegible]
counted))

(+ (count (car x))
(count (cdr x))
1))

~~(else (~~

(set-cdr! x (car (cdr x))))
↳ count ~~[illegible]~~ (cdr x))}
(set-car! x (car (car x))))
↳ (count ~~(cdr~~ (car x)))

something like this