Parser (single quote)
If/let
Scoping?

Parsing (a b (c d))

stack

(

a

b

(

c

d

(c d)

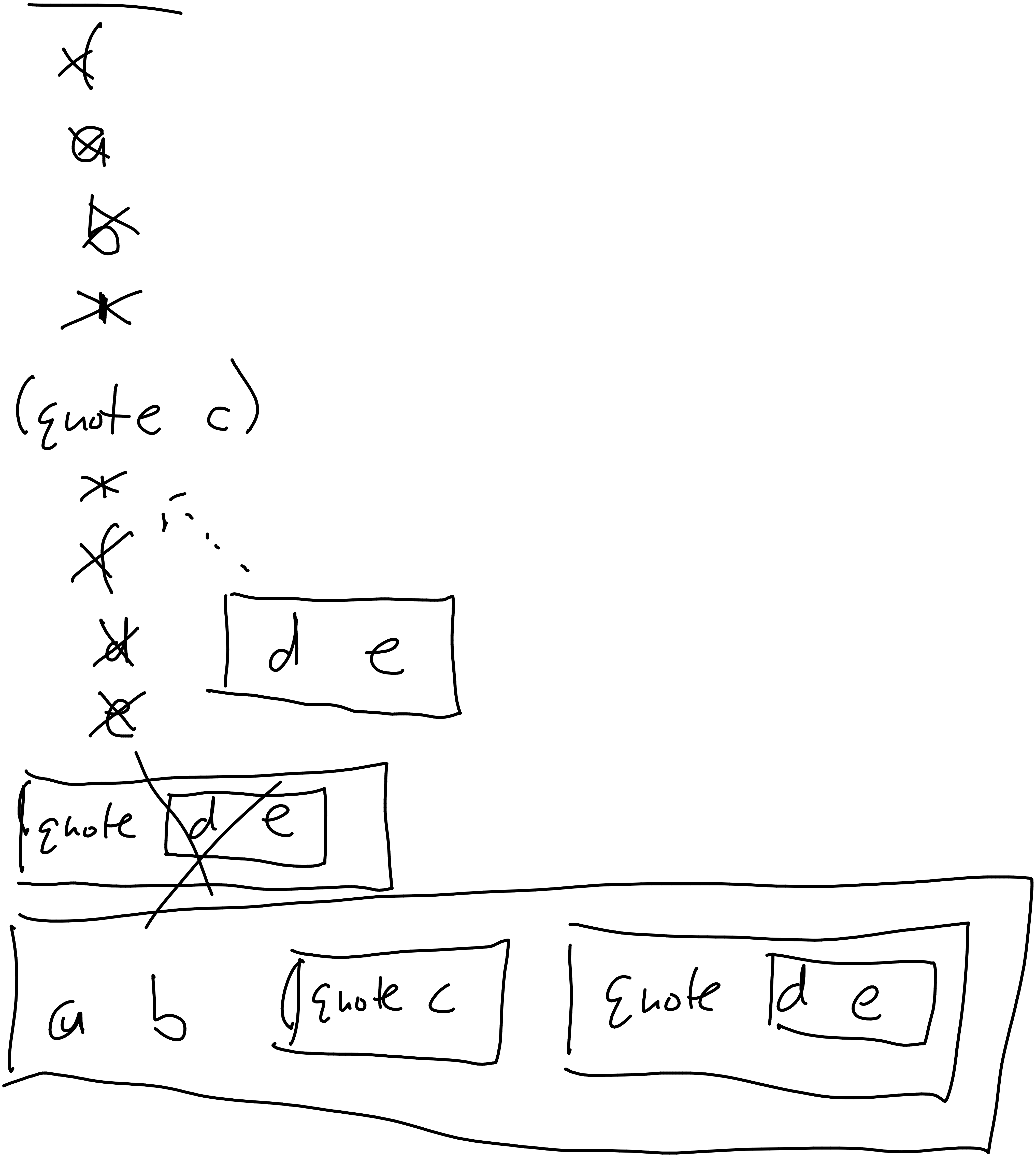
'a (quote a)

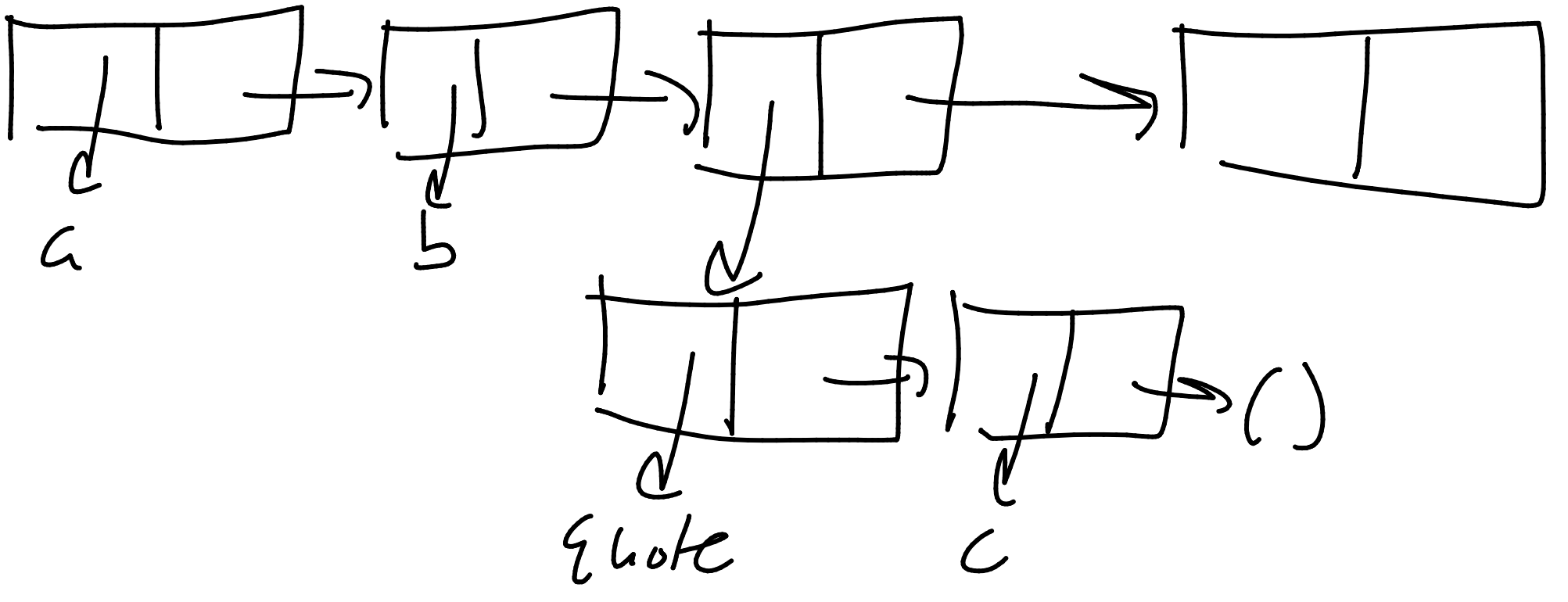
'(a b c) (quote (a b c))

- Proceed as usual so long if the stack doesn't have a single quote `'` on top.
- Also proceed as usual if pushing on a left paren, no matter what the stack looks like.
- In all other cases...
  - Pop the single quote off the stack. Then proceed as if the token sequence had `(quote ....)` wrapped around the value you're pushing. In other words, create a new subtree consisting of `quote` and the new token, and then push that subtree onto the stack instead of the original value you were going to push.

(a b 'c '(d e))

stack





Parse output:

(a b (quote c) (quote (d e)))

Part 5 - if / let

(if #t 3 5) ⇒ 3

(if #f 3 5) ⇒ 5

(if #t
  (if #f 7 9) ⇒ 9
  (if #t 6 3))

let - local variables

(let ((x #t) (y 5) (z 9))
  (if x y z)) ⇒ 5

```
(define C
  (lambda (p)
    (let ((x 4))
      (p 2))))

(define A
  (lambda ()
    (let ((x 3))
      (let ((B (lambda (y)
                 (+ y x))))
        (C B)))))

(A)
```

p is that function again

call function w/ param of 2

x → 4

x → 3

B → function

?

?!

↑ function

(lambda (y)
  (+ y x)
  2 ↖ ?

Which x?

- the x that was visible when the fn was created?
- the x that was visible when the function was called?

```
(let ((x 5))
  (+ x 3))
```

list of local vars and values

some kind of expr

The return value for that expr is the return value for let

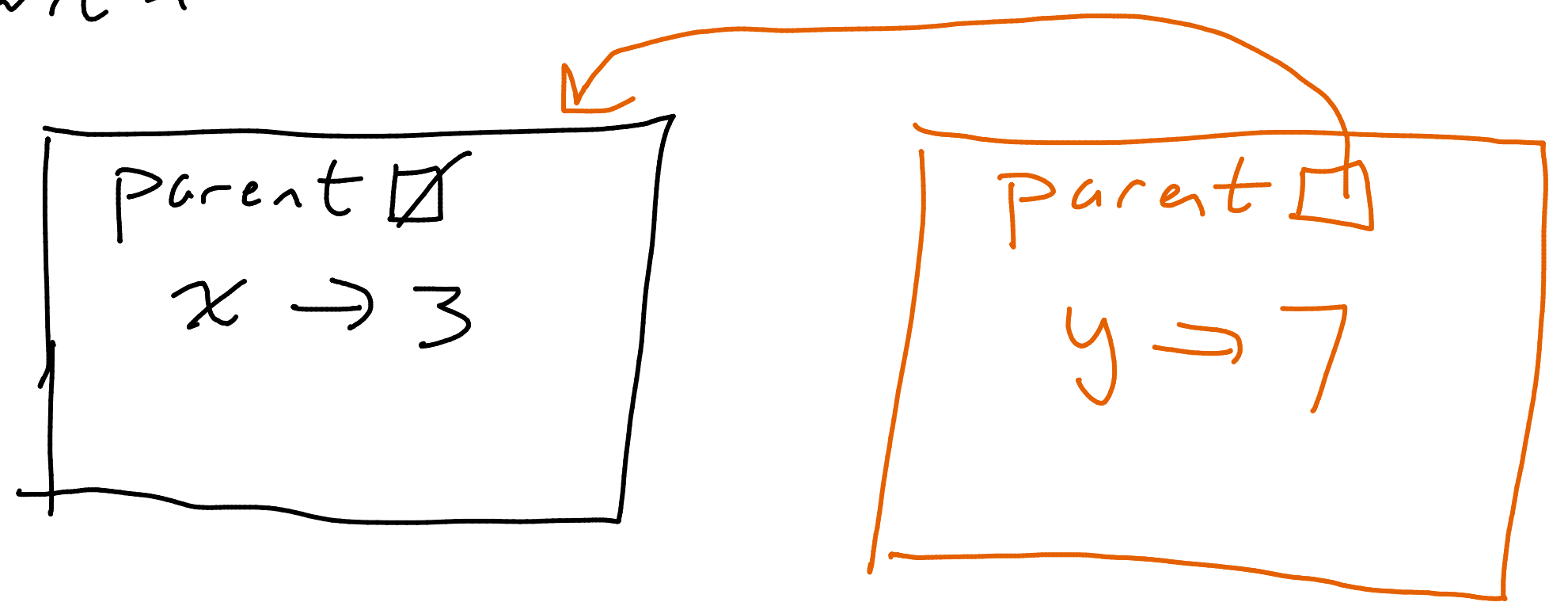
How do you actually make <u>let</u> work in Scheme?

let creates a struct called a frame.

- local vars and what they are bound to
- pointer to the frame that was active when this new frame was made ("parent")

```
(define x 3)
(let ((y 7))
  (+ x y)) → 10
```

Initial global frame that we start with



in the context of this new frame evaluate

(+ x y)

look in Parent (all the way up if necessary) until you find an x

3

7

```
(let ((x 3) (y x))
  5)
```

This causes an error.
The right sides of each pair are evaluated in the parent frame