(define a (f 3 4))

"THE Scheme parsing algorithm"

Start with a stack

Read tokens in order

Push them onto stack unless you hit)

If you hit a)

make a subtree,

start popping off stack and adding to subtree, until you hit a (.

push subtree on stack

Continue

Now, doing last subtree (just another list)

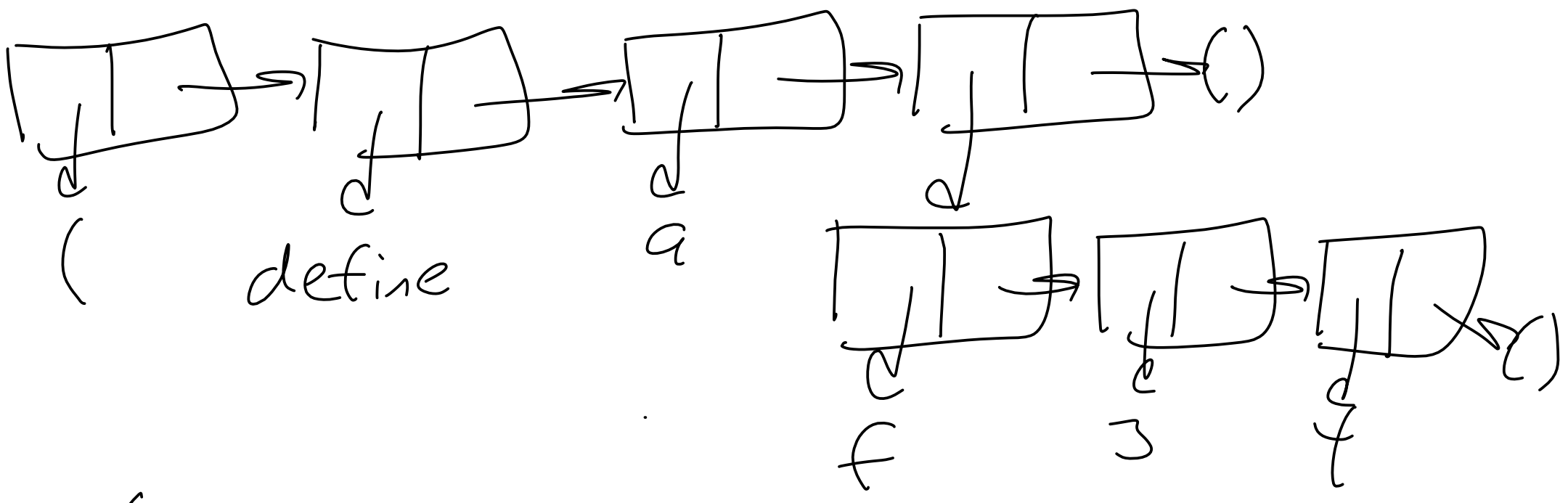
stack: (define a ~~(f 3 4~~

subtree

f 3 4

Now, stack: (define a f 3 4

At this point in time



One last)

new subtree,

define a | f 3 4

stack

define a | f 3 4

curly brace.

```
(define thing
  (lambda (x)
    (+ x 1)))
```

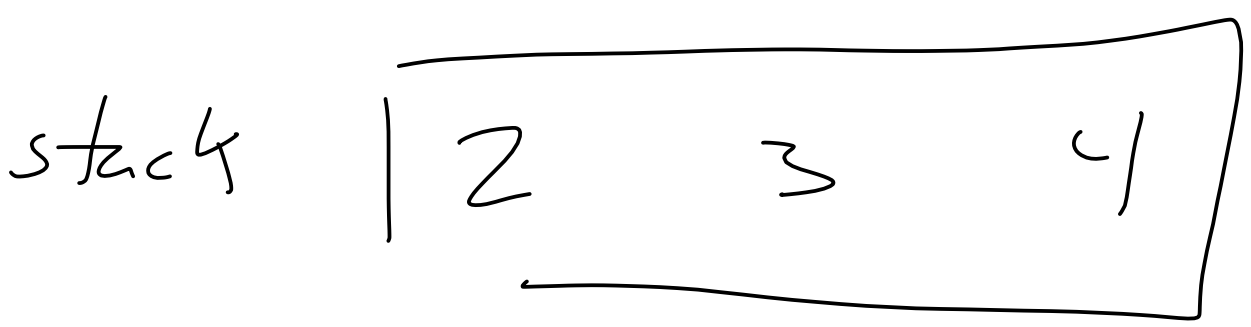
close all open parens.

Scheme has a special case at the top level of a program.

```
(
  (define x 3)
  x
  (define y 7)
  (+ x y)
)
```

```
(define inc
  (lambda (x)
    (+ x 1)))
(inc 3)
```

Parse (2 3 4)



S-expression

A Scheme program is a list of S-expressions

In your code, you will have a list of trees (which are just nested lists)