How Apply() works:

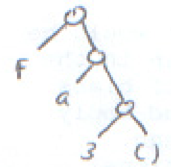
[illegible]

(define (f x y) (+ x y))

(define a 5)

[illegible]

(f a 3)



Don't use set-car or set-cdr except in ~~Built-in~~ Closure

(define foo '(f a 3))

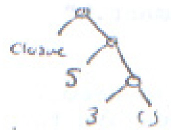
(eval foo (interaction-environment))

what is called?

① Cons::eval

calls → Regular::eval

- evaluate all elements in the list
→ want to construct new tree:

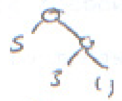


→ helper function → "eval_list()" to do this

```
(define (eval-list L env)
   ~~(map (lambda (x) (eval~~
   (if (null? L) '()
else →   (cons (eval (car L) env)
              (evallist (cdr L) env)))
```

→ take car → Closure

+ cdr, and call apply on closure/Builtin



→ Closure → apply ( )

- Built-in apply must extract arguments, Store them, and construct new treenode.

→ • ~~Closure::apply does~~
Closure::apply ~~create new env~~
new Environment ( ) — ptr in Closure (env from Closure)
→ inside this env, define all params (extract from list → make helper fxn)

→ eval ~~[illegible]~~ body ← can be sequence of expressions, return value of last exp. (helper fxn)

body → apply

```
(define (evalbody L env)
   (car (reverse (evallist L env))))
```