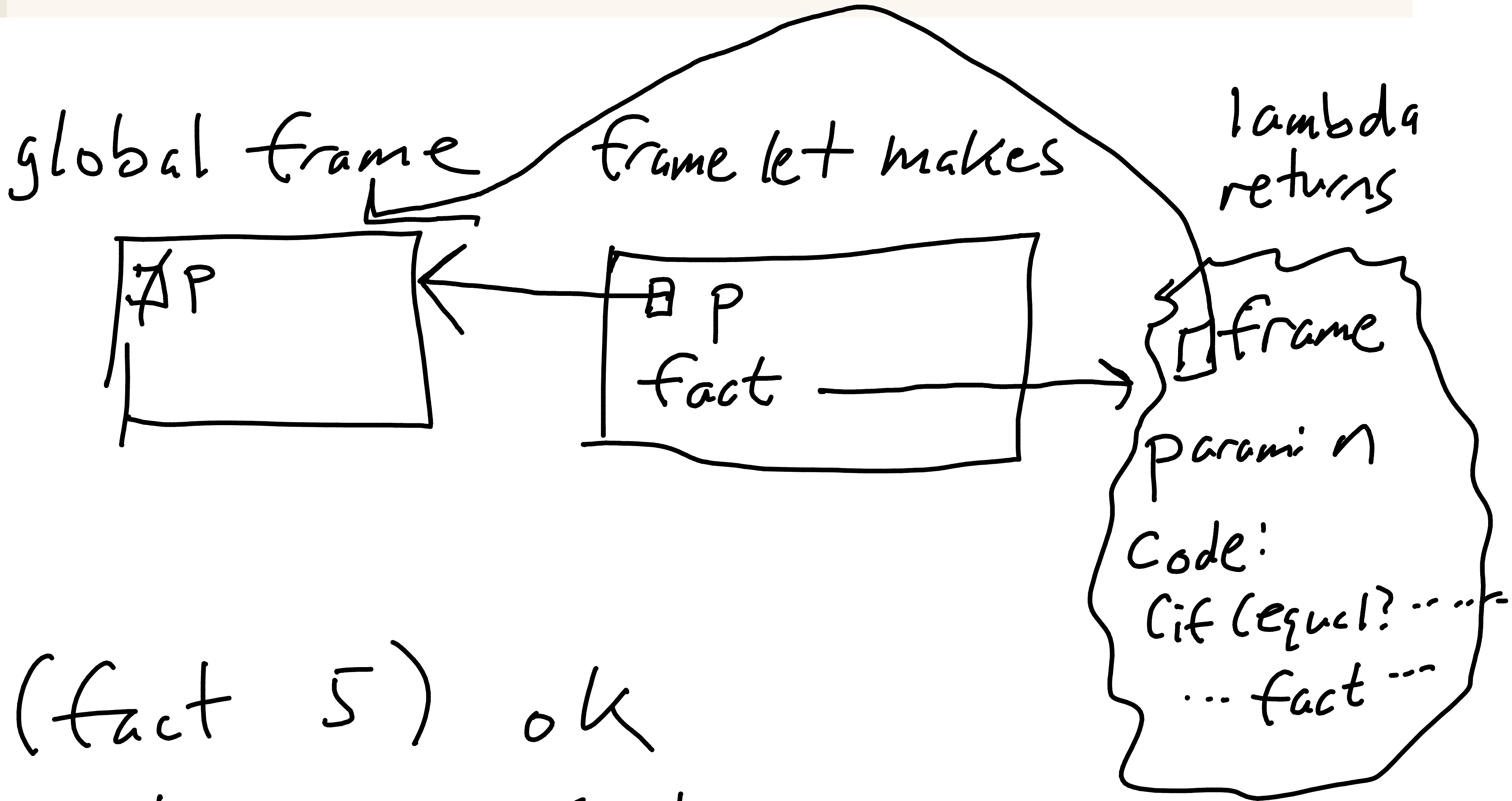
Last assignment - letrec

Knuth's test

RHS of let bindings eval in not the new frame

```
(let ((fact (lambda (n)
              (if (equal? n 1)
                  1
                  (* n (fact (- n 1)))))))
  (fact 5))
```

look in frame let ran in



(fact 5) ok

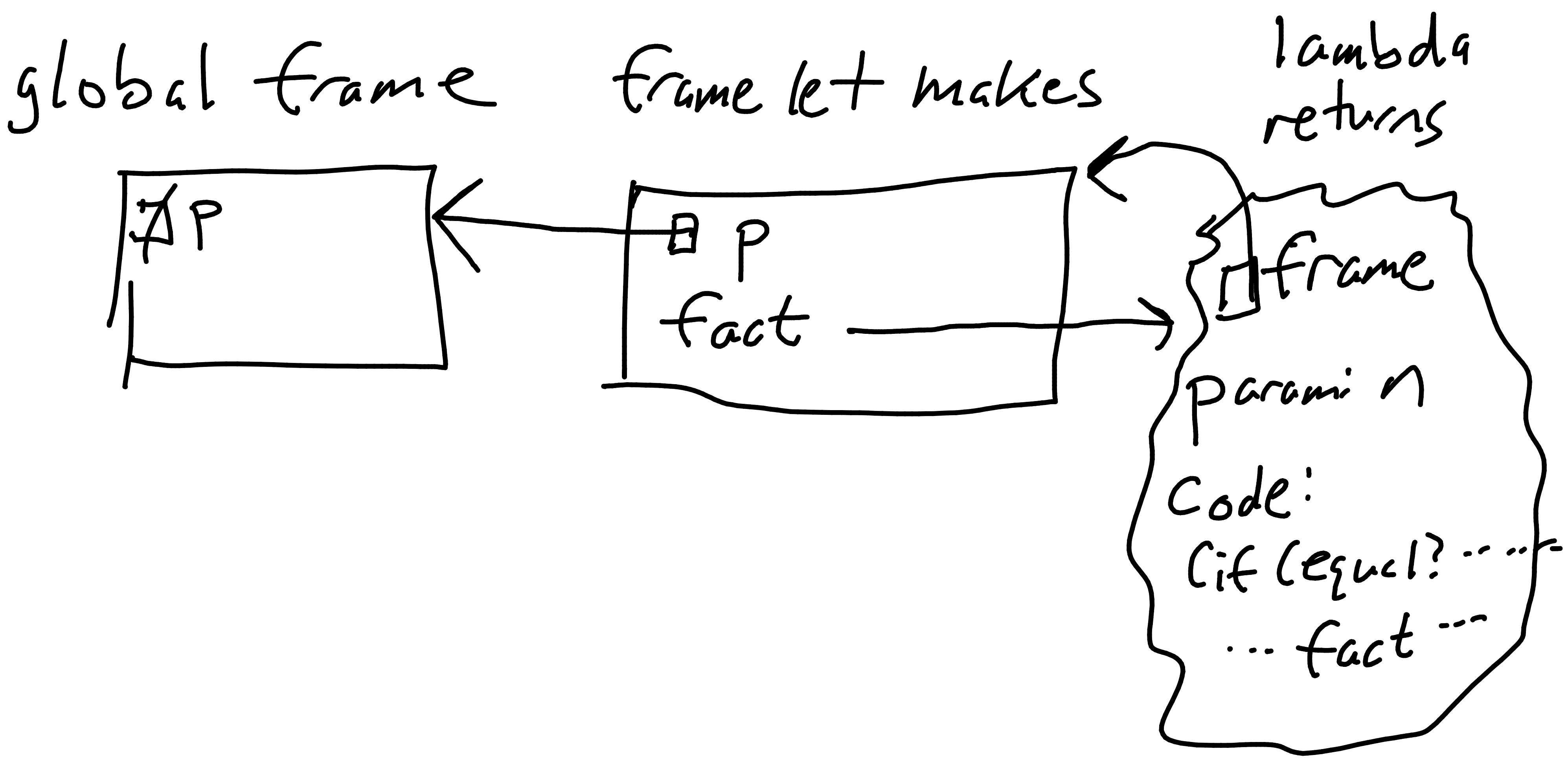
base case fails

(* n (fact (- n 1)))

can't find a binding

```
(letrec ((fact (lambda (n)
                 (if (equal? n 1)
                     1
                     (* n (fact (- n 1)))))))
  (fact 5))
```

Oversimplified: letrec is the same as let, but it evaluates the right side of the bindings in the new frame. (not quite true)

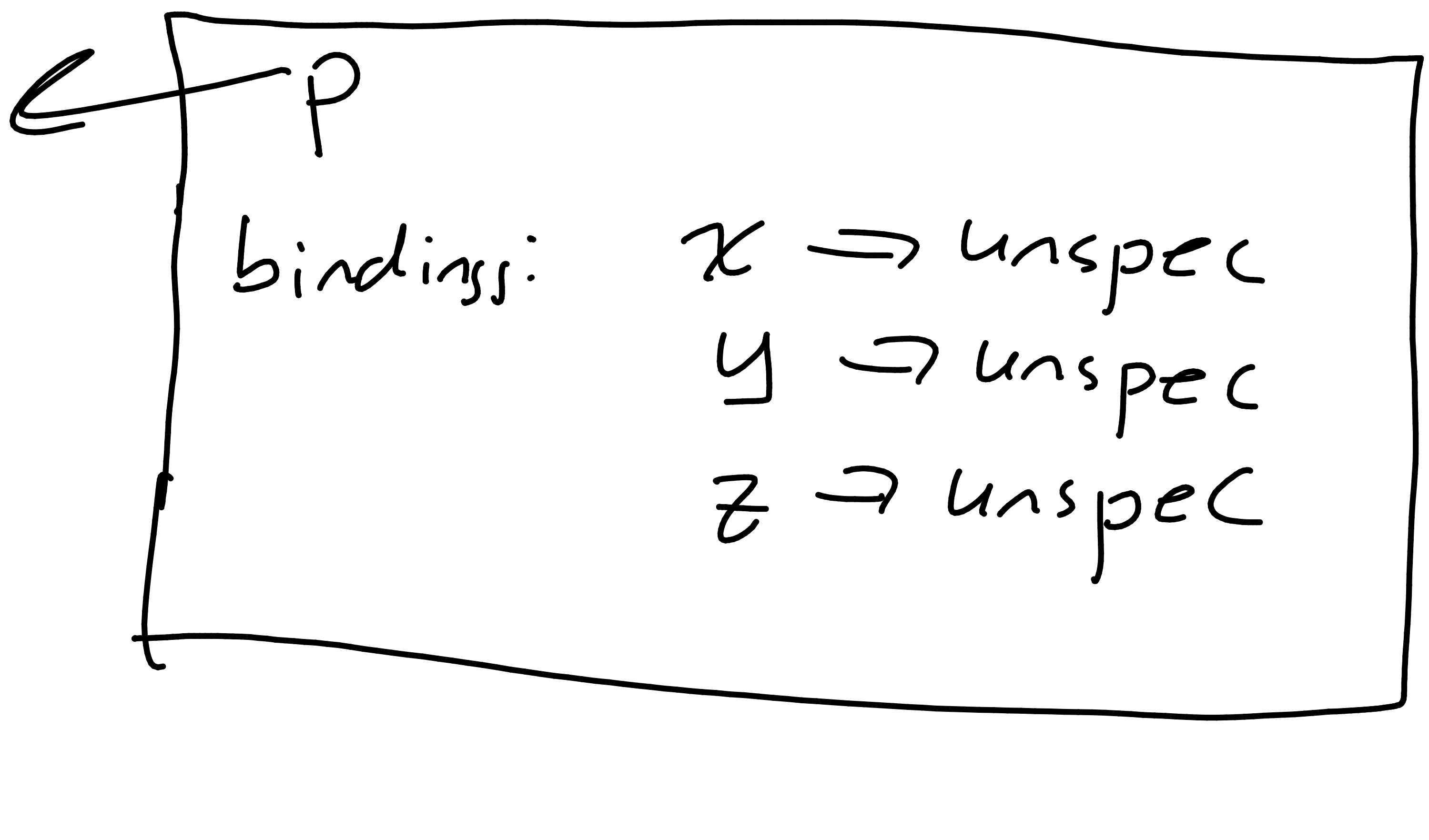


More precisely, what does letrec do?
(also in assignment, also in Scheme ref book)

```
(letrec ((x 1) (y (+ z 3)) (z x) ...)
    ...)
```

① Create a new frame, just like let. Parent is active frame, just like let

② Create each binding w/ a value of "unspecified"



③ Evaluate the right sides *in the new frame*. If any of those evals try to use an unspec, throw error.

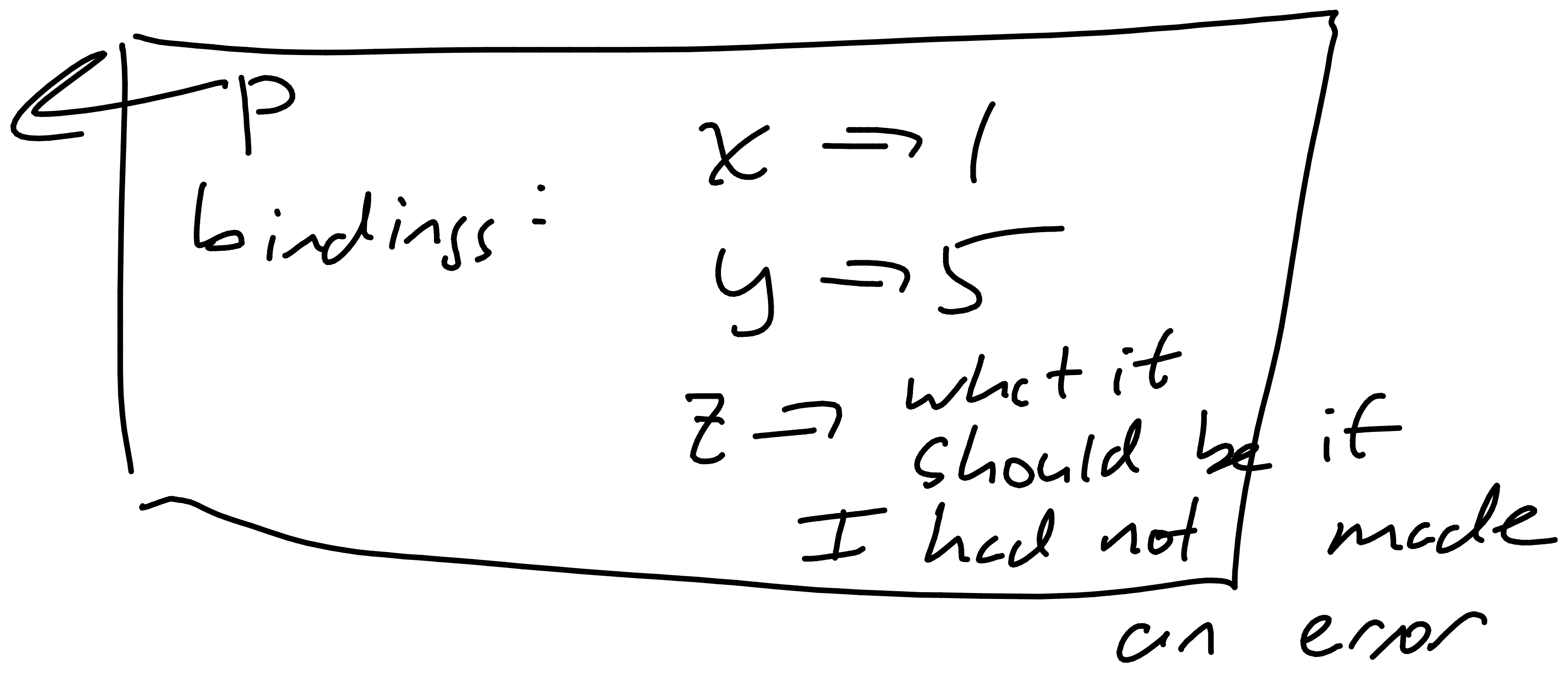
in example:

$1 \rightarrow 1$

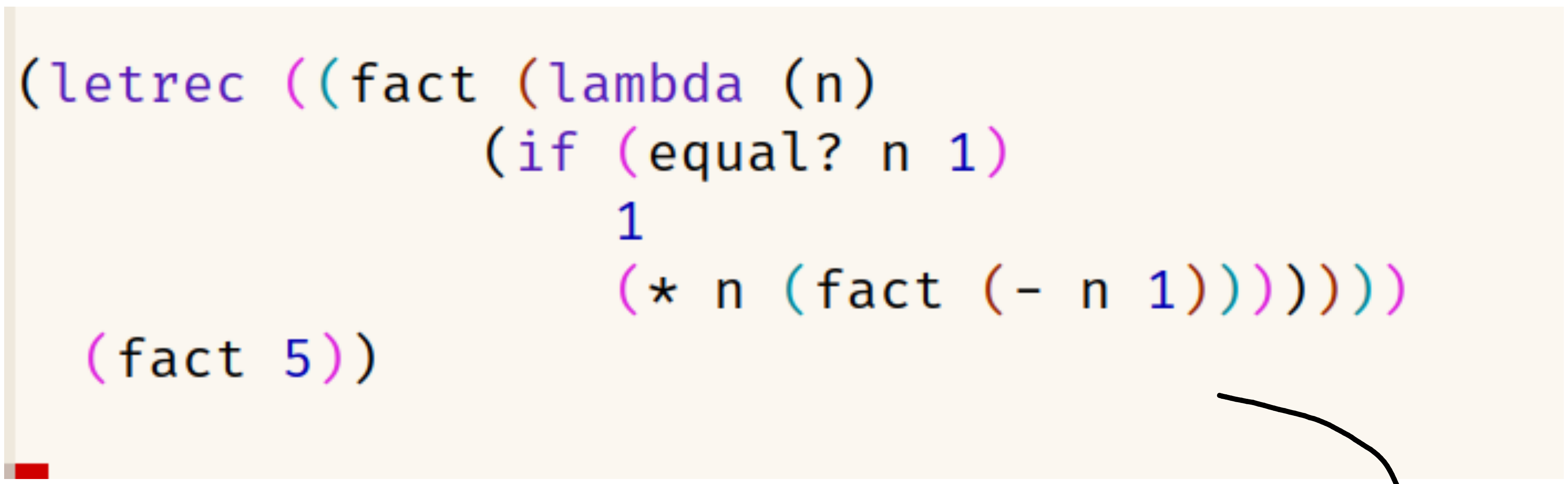
$(+\ 2\ 3) \rightarrow 5$

$x \rightarrow$ unspec $\rightarrow$ error

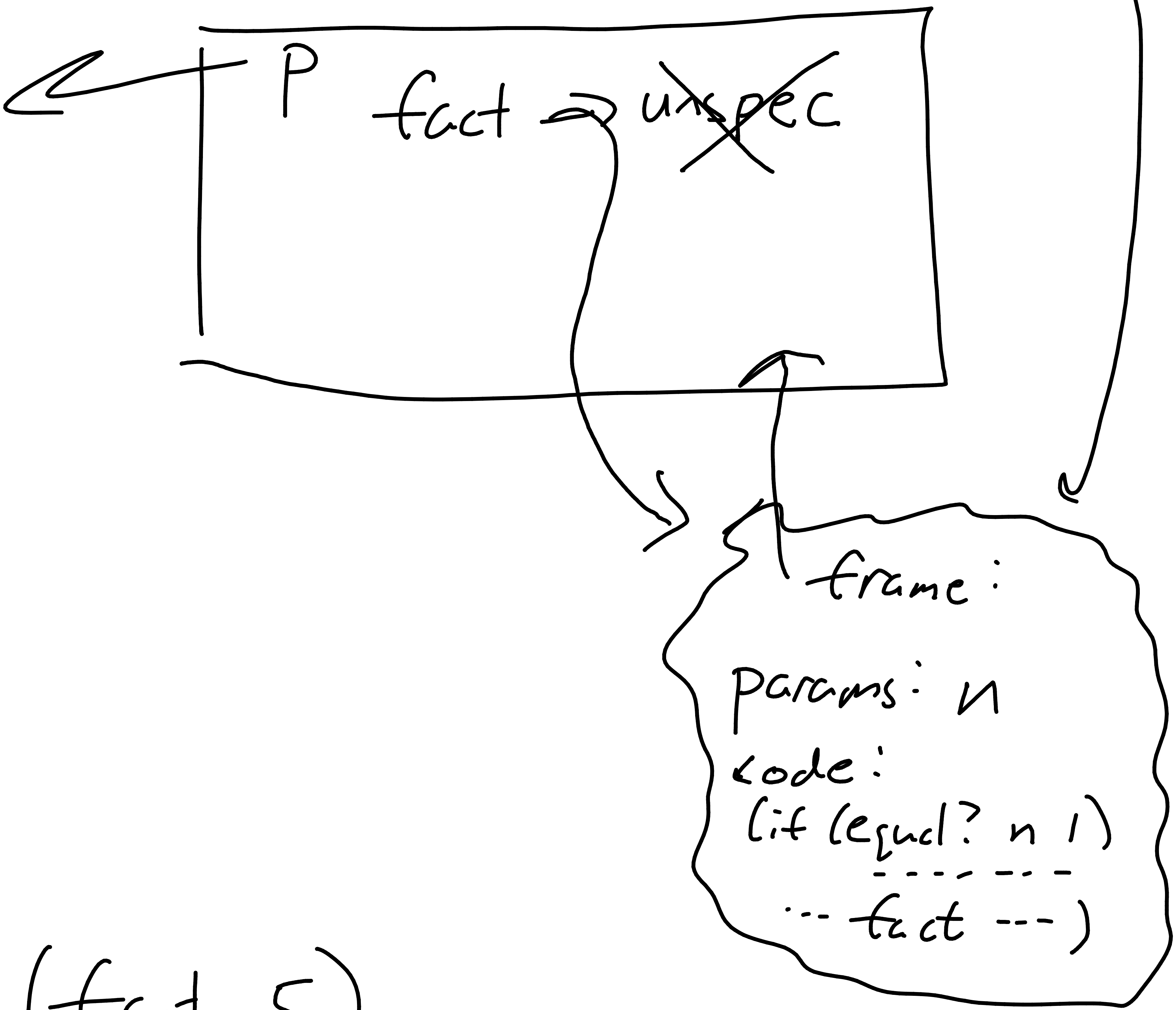
④ If succeed, replace bindings with values from above



⑤ Run the code inside the letrec as usual



```
(letrec ((fact (lambda (n)
                 (if (equal? n 1)
                     1
                     (* n (fact (- n 1)))))))
  (fact 5))
```



(fact 5)

Measuring timing

```
a= get time
   code
b=get time
print(b-a)
```

move around and narrow down

```
SchemeVal *eval(          ) {
    a= get time

    b=get time
} tottime = tottime + (b-a)
```

eval is called from lots of places

global vaiable:

tottime = 0

#include <sys/time.h>

a = gettimeofday($\nearrow$, NULL)

struct
time val