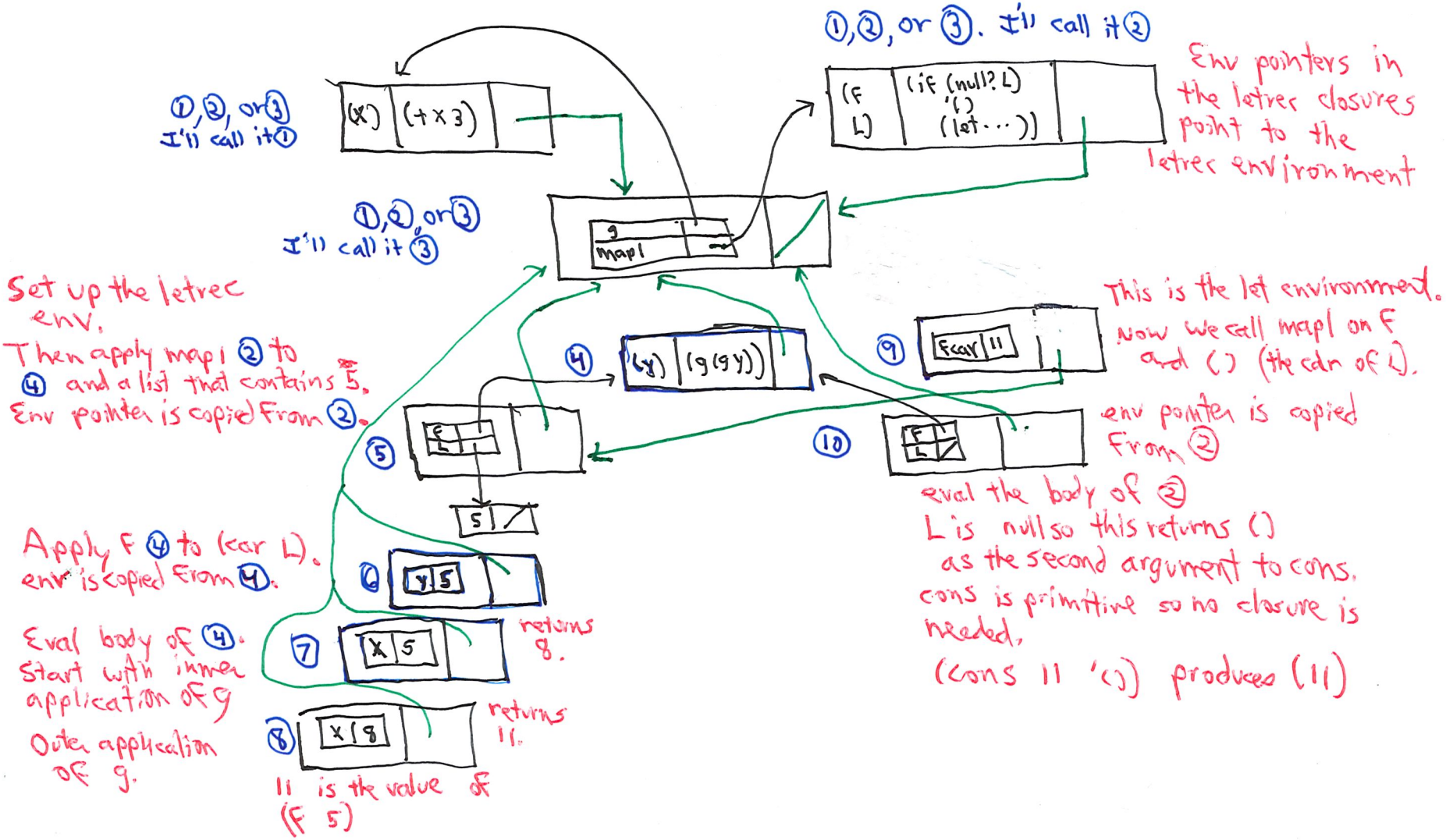
①, ②, or ③. I'll call it ②
Env pointers in the letrec closures point to the letrec environment
①, ②, or ③ I'll call it ①
①, ②, or ③ I'll call it ③
(x) (+ x 3)
(F L) (if (null? L) '() (let ...))
g map1
Set up the letrec env.
Then apply map1 ② to ④ and a list that contains 5.
Env pointer is copied From ②.
④ (y) (g (g y))
⑨ F cdr 11
This is the let environment. Now we call map1 on F and () (the cdr of L).
⑤ F L
5
env pointer is copied From ②
⑩ F L
eval the body of ②
L is null so this returns ()
as the second argument to cons.
cons is primitive so no closure is needed.
(cons 11 '()) produces (11)
Apply F ④ to (car L).
env is copied From ④.
⑥ y 5
Eval body of ④.
Start with inner application of g
Outer application of g.
⑦ x 5
returns 8.
⑧ x 8
returns 11.
11 is the value of (F 5)