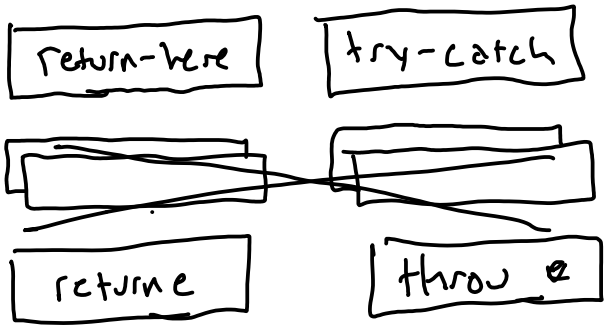


⇒ V

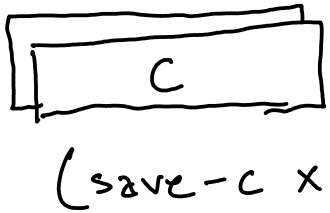
store C

V

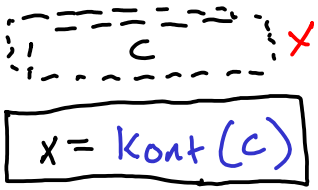
(save-c x e)

kont(c)

(restore-c e e)

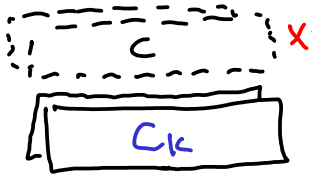


⇒



→ don't include these

e



→ don't include these

⇒

V_a

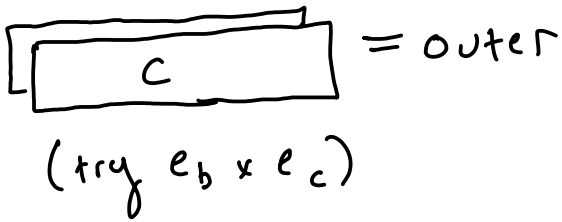
V_a

$V_k = kont(C_k)$

(try e_b x e_c) ⇝ (save-c outer
(let (x (save-c throw-here
(restore-c outer e_b)))
(restore-c outer e_c)))

(throw e) ⇝ (restore-c throw-here e)

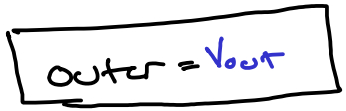
if e_b throws, this never runs



```
(+ 7
  (try (+ 1 (throw 2)) n (+ 3 n)))
```

```
(+ 7
  (save-c outer
    (let (n (save-c throw (restore outer (+ 1 (restore-c throw 2)))))
      (restore outer (+ 3 n)))))
```

⇒

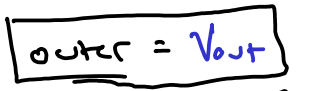


$V_{out} = kont($  $)$

(save-c outer ...)

(let (n ...) ...)

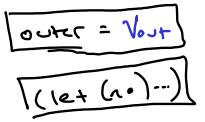
⇒

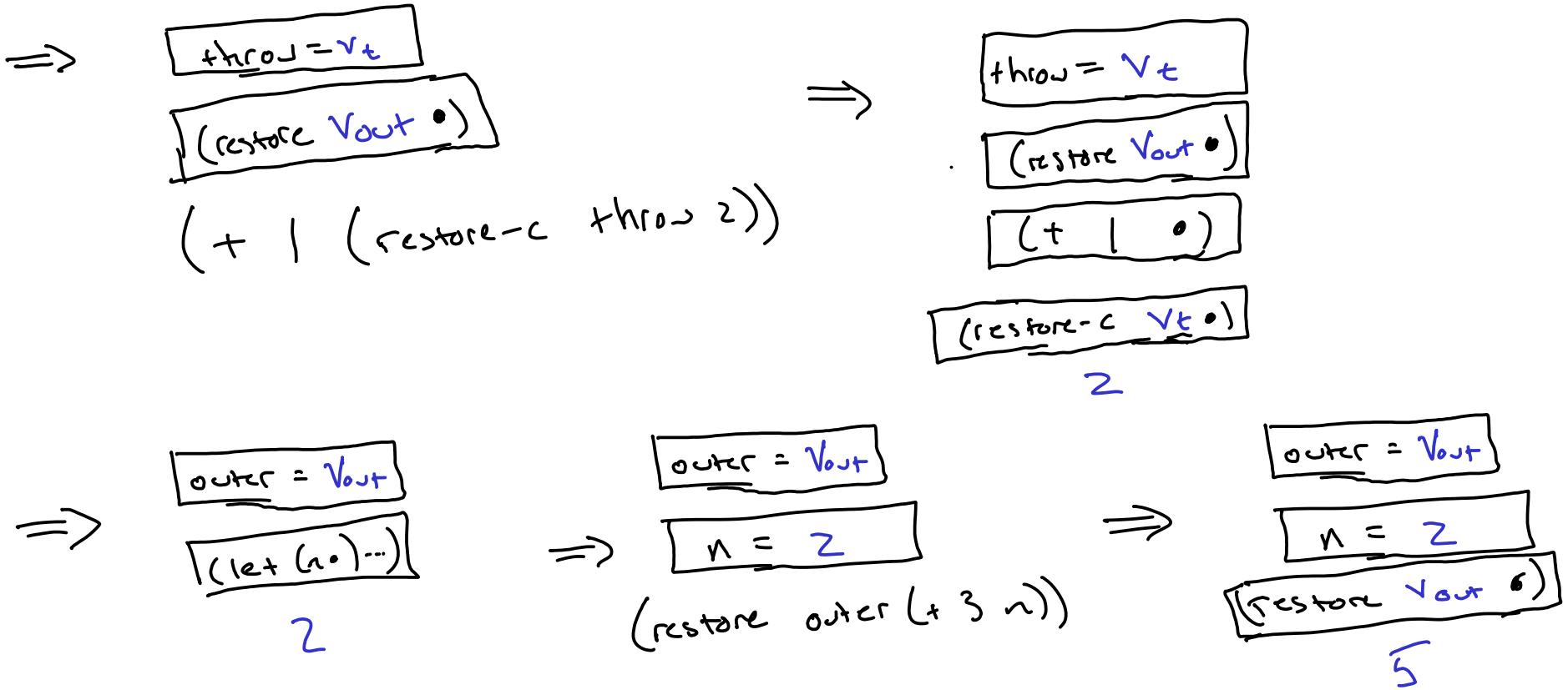


(save-c throw ...)

⇒

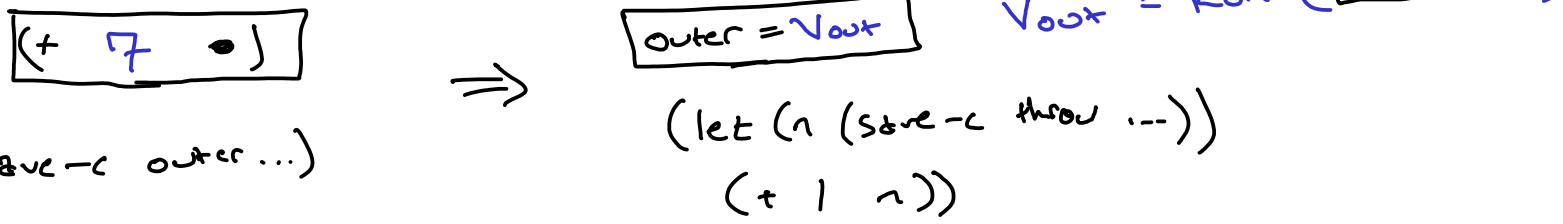
(restore outer ...)

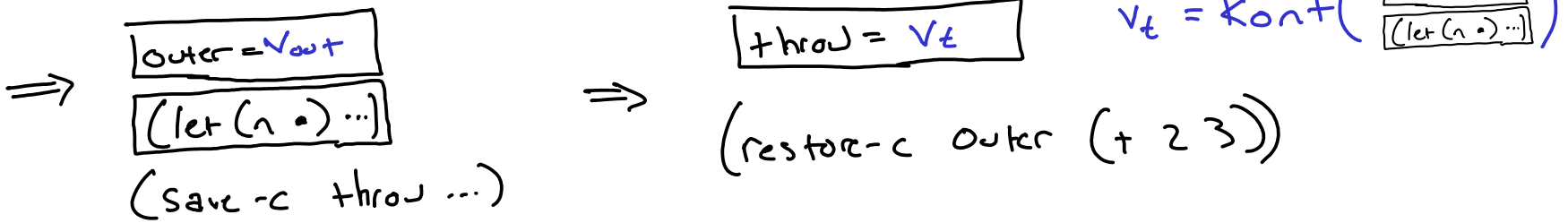
$V_t = kont($  $)$

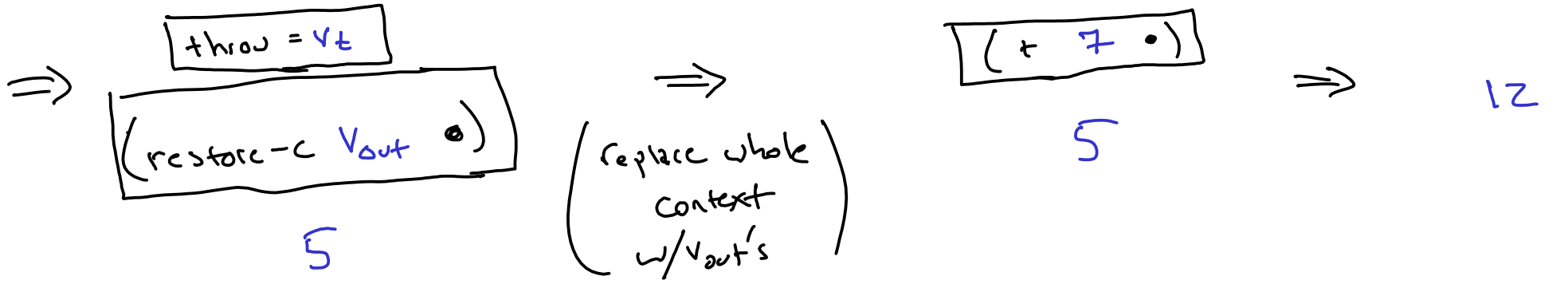

throw = Vt
(restore Vout •)
(+ 1 (restore-c throw 2))
throw = Vt
(restore Vout •)
(+ 1 •)
(restore-c Vt •)
2
outer = Vout
(let (n •) ...)
2
outer = Vout
n = 2
(restore outer (+ 3 n))
outer = Vout
n = 2
(restore Vout 6)
5
(+ 7 •)
5
12

```
(+ 7 (try (+ 2 3) n (+ 1 n)))
```

```
(+ 7 (save-c outer
       (let (n (save-c throw (restore-c outer (+ 2 3))))
         (+ 1 n))))
```






throw = Vt
(restore-c Vout •)
5
(replace whole context w/Vout's)
(+ 7 •)
5
12

```
fun save-c(f): call-cc(f) end
fun restore-c(k, v): k(v) end

fun find-in-list(l :: List, n :: Number):
  save-c(lam(return-here):
      for each(elt from l):
        print(elt)
        when elt == n:
          restore-c(return-here, true)
        end
      end
      restore-c(return-here, false)
    end)
end
```