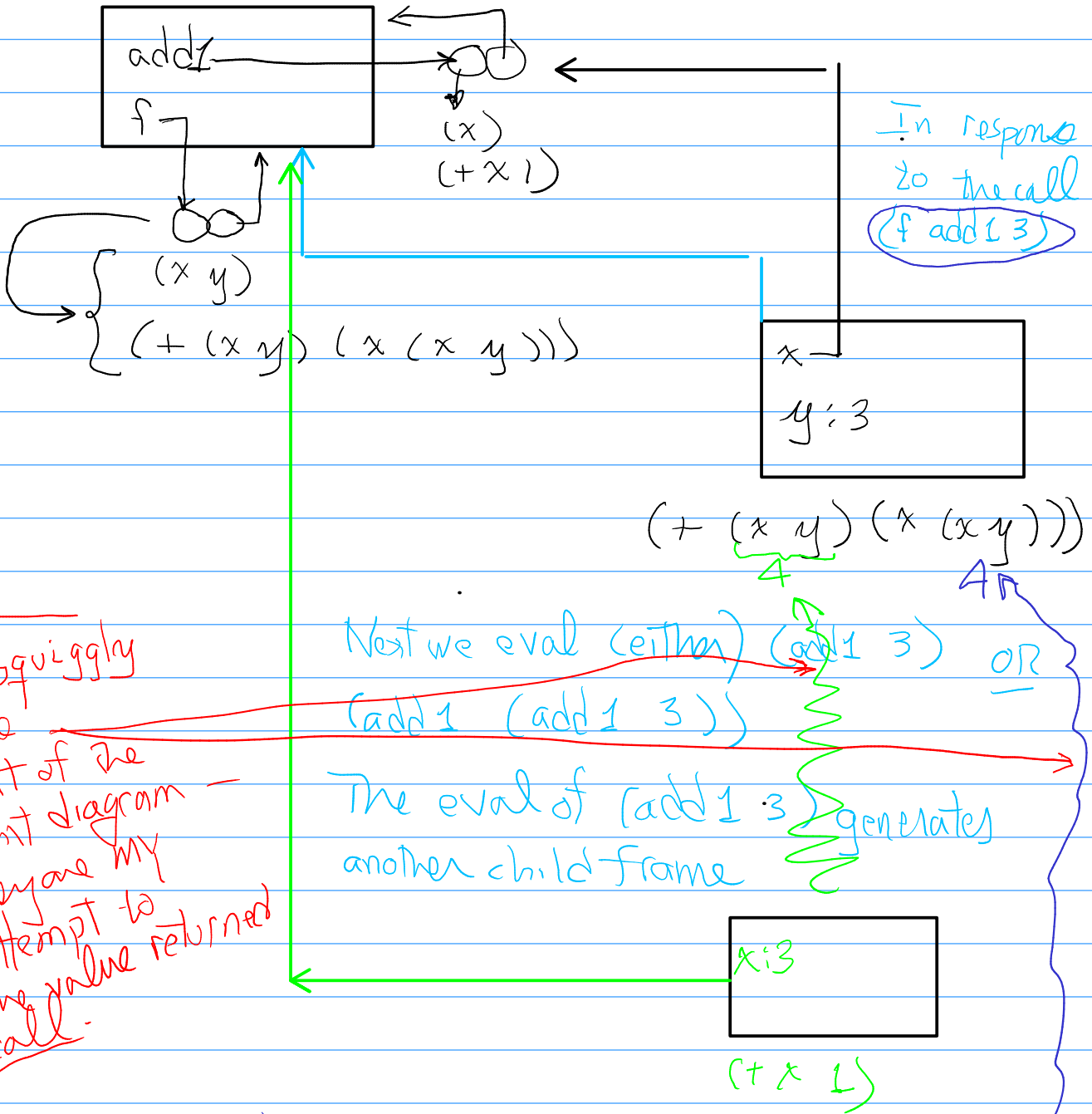
CSc 335 - Office Hour 2.14.24

First Question



Perhaps GC intervenes here, to remove this last frame.

The same sequence of steps now again evals (x y), giving 4 — creating another frame —

Perhaps GC turns on again

And finally a similar sequence evals (x 4)
... yes, another child frame!

2ND Question

Recursive soln to the zero counting problem where we (this time) count only the zeros occurring on the right side of the number.

Is this a decent spec? The first thing to look at might be whether the spec is complete — is the input sufficiently restricted and is the output adequately described?

Input: NO not enough — we should probably add that the input is a Scheme number [so 0000 is not allowed, and neither is 00102, etc: No leading zeros]. In addition, it might make sense to restrict attention to non-negative integer inputs — the idea being that negative inputs could be handled with a wrapper. Finally, 'number' is impossibly vague — we want integer inputs.

Output: do we understand 'right side of a number'? Better: count the zeros to the right of the rightmost non-zero digit

Perhaps we'll notice something more later — frequently happens — but this seems to be enough to start.

Divide & Conquer Idea? Termination Idea?

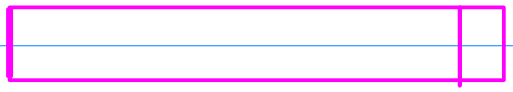
We can make progress one digit at a time — working from the right

using modulo 10

Remaining work

using quotient 10

D&C



(quotient n 10) (modulo n 10)

"smaller problem of the same kind"

Termination Idea

Continue so long as (modulo n 10) = 0

WAIT!! This termination condition will fail for input 0.

So: we'd better start with $n \geq 1$

etc.

So let's try it

```
;; n >= 1 is a scheme integer
(define (count-rightmost-zeros n
  (cond ((not (zero? (modulo n 10))) 0)
        (else (+ 1 (count-rightmost-zeros
                    (quotient n 10)))))))

;; post condition
   as above
```