Goals for today: look at things you can do when functions are data (you can pass functions to/from functions!)

"partially run a function on only some of its parameters, and finish later."

f(—, —, —, —)

lots of work happens based on first 3 parameters, it might be nice to get that done if I don't know yet what 4th parameter is

↳ fake this with a trick

Currying (Haskell Curry)

idea is to have "nested" functions of one parameter each

```
;; redoit in a curried fashion
(define mult
  (lambda (a)

    ;; return another function of one parameter
    ;; that will multiply the two numbers
    (lambda (b)
      (* a b))

    )
)
```

(define patial (mult 3))

(patial 7) → 21

(patia 4) → 12

Curried function - a function of one parameter that returns another function

Late - you need to be here on time

Use of online resources and/or AIs

Goal of this part of course is to learn how to do functional programming.

Goal is not for you to find code elsewhere that solves problems and submit them to me.

You may not search/ask/etc of tools "how do I <do the assignment>"?

"imp binary search trees in Scheme"

You may ask

"How do I get the second item in a list in Scheme?"

(car (cdr lst))

What about 4 params?

```
(+ 2 3 4 5)
```

```
(define plus4
  (lambda (a)
    (lambda (b)
      (lambda (c)
        (lambda (d)
          (+ a b c d))))))

(define my-ans (plus4 1 2 3 4))

(my-ans 5)
```

Higher order functions

- a higher order function is a function that takes a function as a parameter

Applications - Google does a lot of work with tasks that occur on multiple computes, and the results get combined

eg. "find all web pages w/ the word 'apple" in them"

Google found themselves reinventing over and over, code that does:

"do a task on multiple machines"

"combine together"

map takes 2 params

- function of 1 param
- a list of data

returns a list that ran the fuction on each item of orig list

---

"reduce" in Scheme is called "fold"

fold-left ✶✓

fold-right

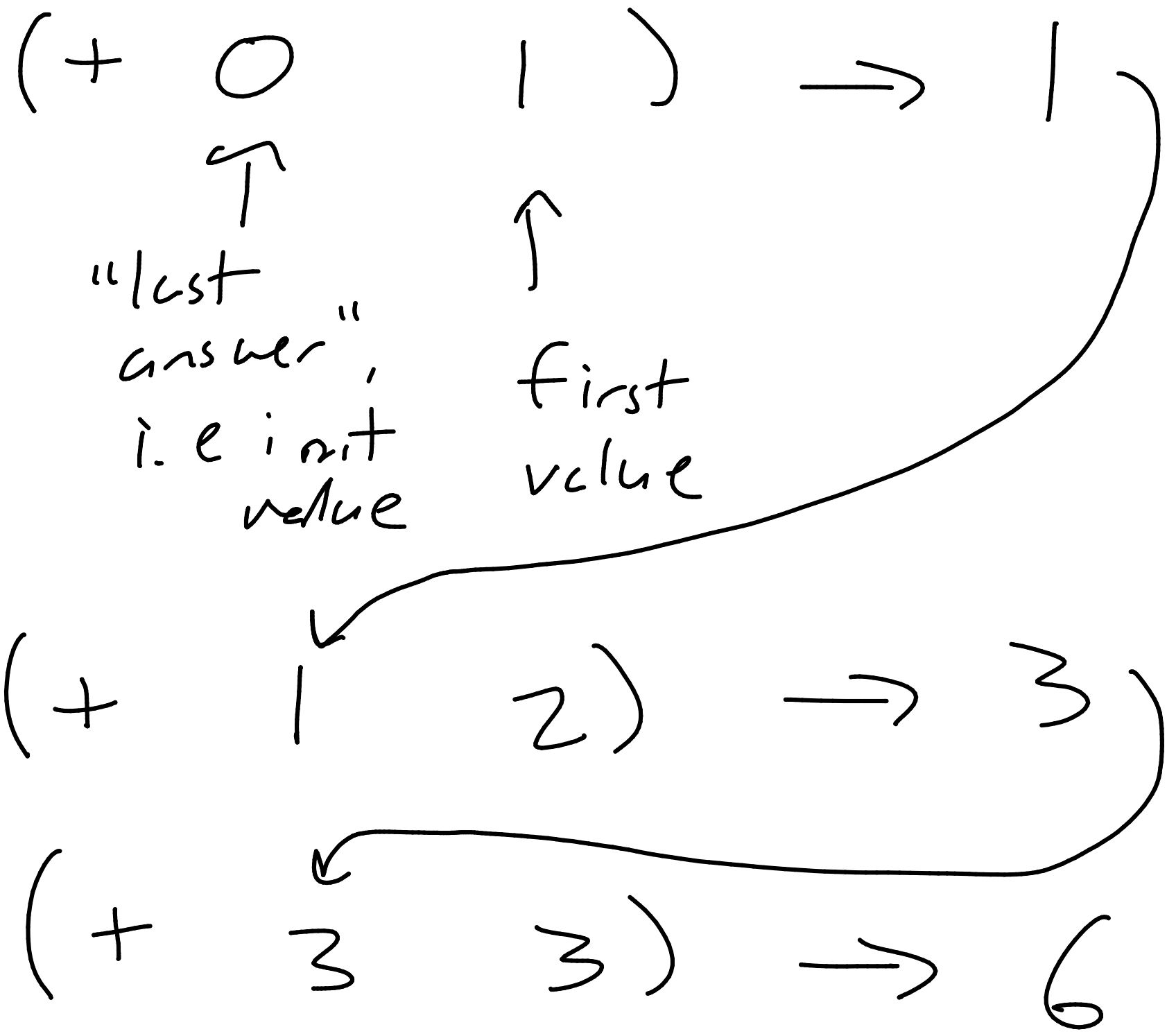
```
(fold-left + 0 '(1 2 3))
```

↑ function of two params

↑ initial value

↑ list of values

Applies the function repectedly to the answer from the previous run and the next parameter


(+ 0 1) → 1
"last answer", i.e init value
first value
(+ 1 2) → 3
(+ 3 3) → 6