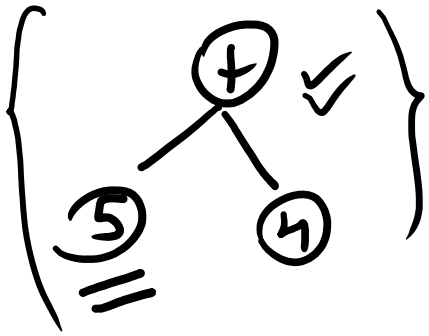
R-0 and ast

17 January 2022 03:00

ASTs in Racket

struct



(struct (Int (value)) ✓ ✓

⟷

(Int 5) (Int 4)

(struct Prim (op args*))

(Prim '+ (list (Int 5) (Int 4)))

Int arithetic — Concrete syntax

exp ::= int | (read) | (- exp) | (+ exp exp)

Ro ::= exp

-> (44) -> (int) -> Exp

-> (+ 10 32)

~~-> (+ (+ (10 (read)) 32)~~ X

-> (+ (+ 10 (read)) 32)

-> (+ (- 5) 10)

-> (- 5 10) X

abstract syntax

17 January 2022 03:22

exp ::= (Int int) | (Prim 'read '()) |
(Prim '+ (list exp exp)) |
(Prim '- (list exp))

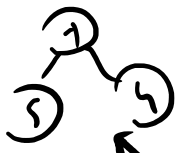
R0 ::= (Program '() exp) body

(struct Program (info body))

LS → AS → m c.

{ Lex, flex, Bison }

Prim '+ (list)



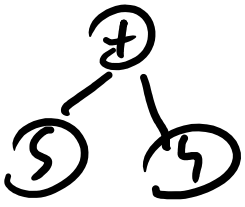
(match e

e ::= (Int 8)

(Int n)

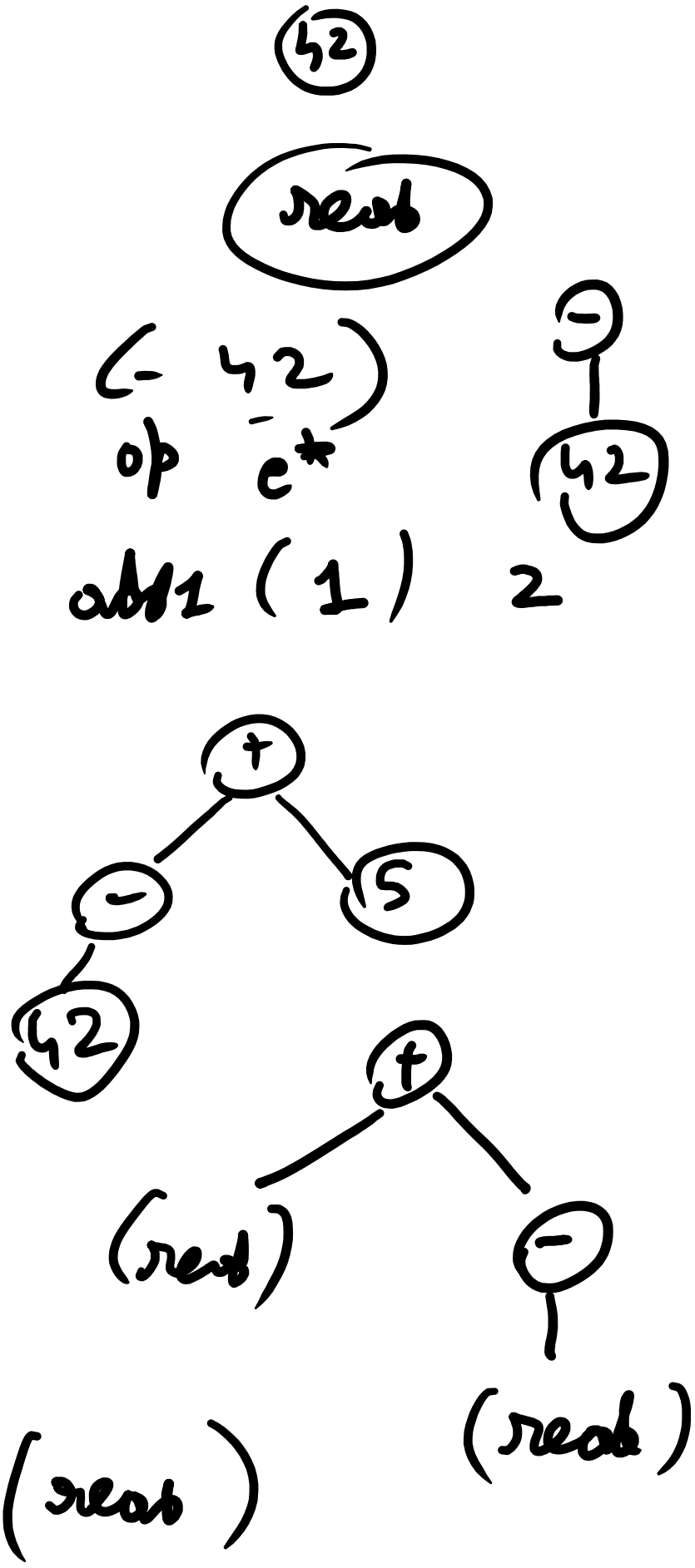
Prim op e*

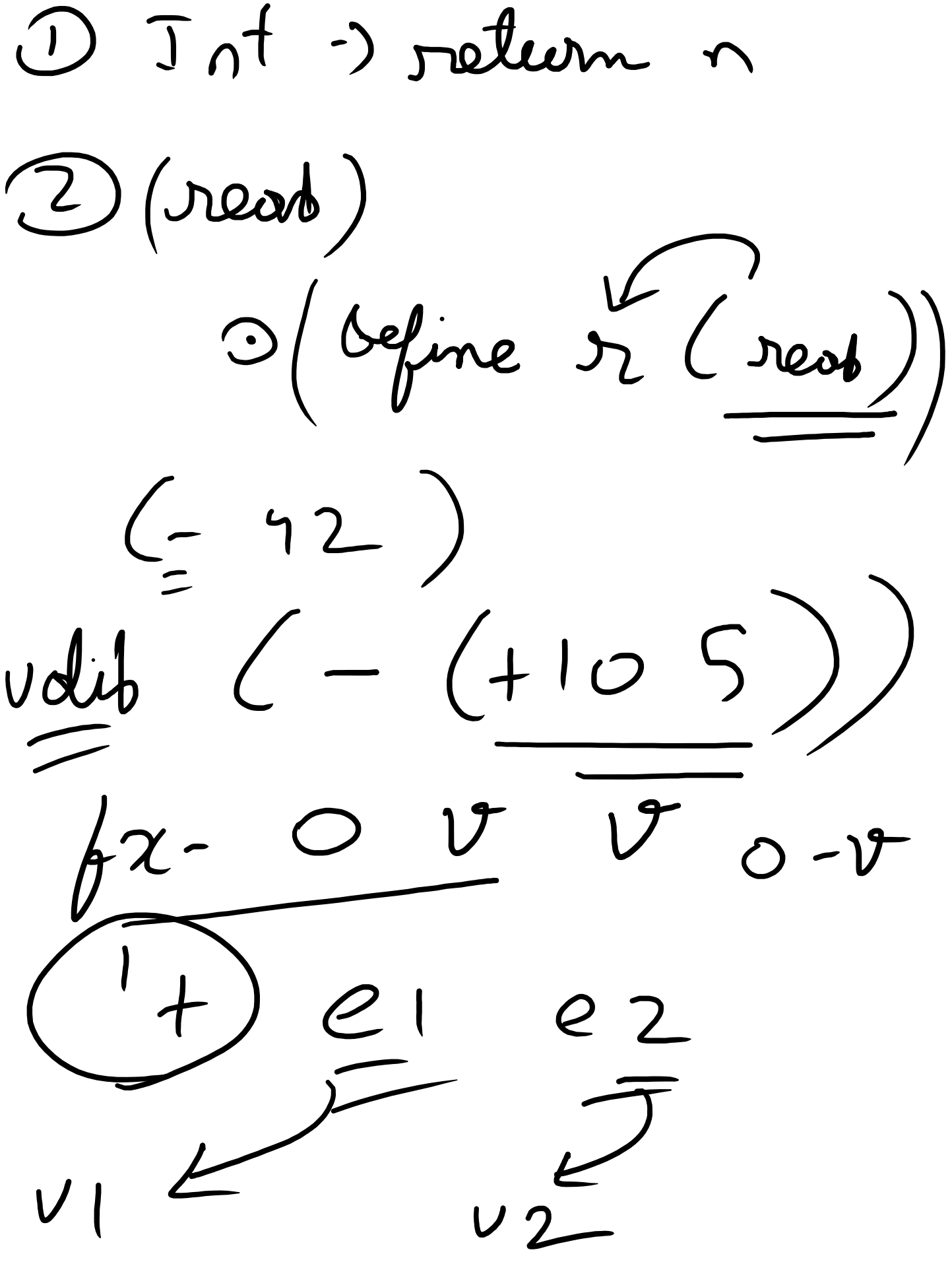
'+ (list (Int 5) (Int 4))



odd 1

(list h1 h2 h3)


42
read
(- 42)
op e*
-
42
obj (1) 2
+
-
5
42
+
(read)
-
(read)
(read)


① Int -> return n
② (read)
(define x (read))
(- 42)
vdiv (- (+ 10 5))
fx- 0 v v 0-v
'+ e1 e2
v1
v2

(L0) V1 + X2

(define x (+ 10 4))

x = 10 + 4

e: (- (+ (+ 1 2) 3))

(Prim' - (Prim' + (. . . .))

① (- (+ (+ 1 2) 3))

'- (list (+ (+ 1 2) 3))

e

1.1 (+ (+ 1 2) 3))

e1 e2

3 3

1.1.1 3

(+ 1 2)

e1 e2

3

1.1.2 (Int 3) 3

interp-exp

RO (+ 10 5)

Compiling