10-1

$R_2 \rightarrow R_3$

$Ty ::= \ldots \mid Unit \mid (Vector\ Ty\ \ldots)$ (Vector Int Bool)

$e ::= \ldots \mid (unit) \mid (vector\ e\ \ldots)$
$\mid (vector\text{-}ref\ e\ number)$
$\mid (vector\text{-}set!\ e\ number\ e)$

```
let x := 17 ;n
    v1 := vector    x    (x+1)    (x ≤ 20)    in
    y := vector-ref    v1    1    in
    z := vector-set!    v1    1    0    in
    z := vector-ref    v1    1    ;n
  y + z
⇒ 18
```

R_3 has automatic memory mangement (i.e. no free)

$\Gamma \vdash (unit) : Unit$

$$\frac{\Gamma \vdash e_v : (Vector\ T_0 \ldots T_i \ldots T_n) \qquad \Gamma \vdash (vector\text{-}ref\ e_v\ i) : T_i}{\Gamma \vdash e_0 : T_0 \quad \ldots \quad \Gamma \vdash e_n : T_n \qquad \Gamma \vdash (vector\ e_0 \ldots e_n) : (Vector\ T_0 \ldots T_n)}$$

$$\frac{\Gamma \vdash e_n : T_i \qquad \Gamma \vdash e_v : (Vector\ T_0 \ldots T_i \ldots T_n)}{\Gamma \vdash (vector\text{-}set!\ e_v\ i\ e_n) : Unit}$$

$\emptyset \vdash (vector\text{-}ref\ (vector)\ 0) : \_$

10-2/ Change 1: Add the option for the random type to be Unit

- (vector)
- (vector int bool unit)
- (vector int x 6)

```
let v1 := vector 0 #t (let v2 := vector 0 1 2 3 4 5 in
                       vector-set! v2 1 (+ 17 read))
in
vector (vector-ref v1 0)
       (< (vector-ref v1 0) (read))
       (vector-set! v1 0 22)
```

#103. big mem N M

= allocate N bytes memory M times
(where only 1 is reachable at once)

- defeat the optimizer

```
v0 = (vector ... (read) ...)
(vector-set! v0 0 (read))
v1 = (vector .... (read) .... (vector-ref v0 □))
:
final-answer = (vector-ref vn)
```

- predict the answer

(- □ prediction)

test = does it return 0?

R_3 optimization

(vector-ref (vector ....) i) ⇒ e_i

known vector

$e_0 \ldots e_i \ldots e_n$

(vector-ref (vector (read) 19) 1) ⇒ seq (read) 19

(vr (vector (read) (read)) 0) ⇒ let ans := read
seq read ans

seq a b = let _ := a in b

seq (num n) b = b

let v1 := (vector ...) in
let v2 := ... in
(vector-ref v1 0)

(vector-set! (if (≤ 17 (read)) v1 v2))

---

typec Γ e ⇒ t instead ⇒ (e : t)

R_3 e := .... | (e : t)

typec Γ (var x) = ((var x) : Γ(x))

---

typep (program ; e) = (program ; [type → e's t] [tyenvp → (var → Ty)] e)