UMPIRE Practice

"Determine if 2 singly linked lists intersect. If they do, return the intersecting node. Intersection is defined by reference, not value."

U: What kind of list? In place? By reference? Size limitations?

- Test cases:
  - l1: 4 → 5 → 2
    l2: 3 → 5 → 2 ⇒ 5 } happy case
  - l1: ∅
    l2: ∅ ⇒ none or ∅ } edge case

M: pointer bookkeeping, two pointer?, multi pass?

P:

```
l1: 4
        \
         5 → 2 → ∅
        /
l2: 6 → 3
```

✗ Idea: pointer bookkeeping - walk pointers P1 and P2 through? But not guaranteed lists are same size

✗ can't work backwards from front easily w/o N^2 passes

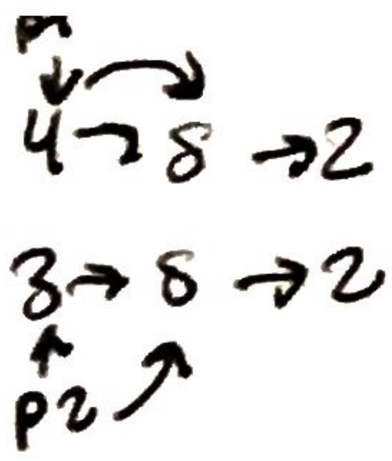
- Multi pass: calculate length of 2 lists & remove prefix of longer list before comparing

①
```
      p1
       ↓
       4 → 5 → 2
6 → 3 → 5 → 2
            ↑
            p2
```

```
if p1 != p2
    return None
n l2 --
n l1 --
```

②
```
      p1
       ↓
       4 → 5 → 2
6 → 3 → 5 → 2
    ↑
```

(can't read his handwriting here :( )

③ 

(also can't read but basically put pseudocode here)

I:
```
def find_intersection(l1, l2):
    # if a list is empty, return
    if not l1 or not l2:
        return None
    p1, nl1 = get_end(l1)  # nl1 = num. of nodes in l
    p2, nl2 = get_end(l2)
    if p1 != p2:
        return None
    # phase ②
    if nl2 > nl1:
        dl = nl2 - nl1   # dl = difference in lengths
        p2 = fforward(l2, dl)
    else:
        dl = nl1 - nl2
        p1 = fforward(l1, dl)
    while p1 != p2:
        p1 = p1.next
        p2 = p2.next
    return p1

def get_end(l):
    n = 1
    p = l
    while p.next:
        p = p.next
        n = n+1
    return p, n

def fforward(l, d):
    p = l
    while d > 0:
        p = p.next
        d = d-1  #R
    return p
```

R: debug line-by-line using watchlist

l1: 4 → 5 → 2
l2: 6 → 3 → 5 → 2

---

p = ~~4~~ ~~5~~ 2
n = ~~4~~ 2 3 get_end()

---

p1 = (2)
nl1 = 3
p2 = (2)
nl2 = 4

dl = 1

---

l = (6) ffoward()
d = ~~1~~ 0
p = ~~(6)~~ (3)

---

p2 = (3) (5)
p1 = ~~(4)~~ (5)

---

return (5)

- may want to debug using other test cases too → run through <u>each</u> line of code at least once

E: evaluate

- could improve ffoward
- Time/space:
  - get_end: O(N)
    ffoward: O(N)
    find_intersection: O(N)
    O(4N) = $\boxed{\text{O(N) time}}$
  - p1, p2, nl$_x$, dl ⇒ $\boxed{\text{O(1) space}}$