Check for index i
to i+1

((= i (vector-length vec)) true)

```
(: vector-eq? : (Vectorof Integer)
                (Vectorof Integer)
                -> Boolean)
(define (vector-eq? vec1 vec2)
  (local
    {
    (: helper : (Vectorof Integer) (Vectorof Integer)
                Integer -> Boolean)
```

```
(define (helper vector1 vector2 idx)
  (cond
    [(and (= idx (vector-length vector1))
          (= idx (vector-length vector2))) #t]
    [(= idx (vector-length vector1)) #f]
    [(= idx (vector-length vector2)) #f]
    [else (if (= (vector-ref vector1 idx)
                 (vector-ref vector2 idx))
              (helper vector1 vector2 (+ idx 1))
              false)]))
(helper vec1 vec2 0))
```

Both have reached the end

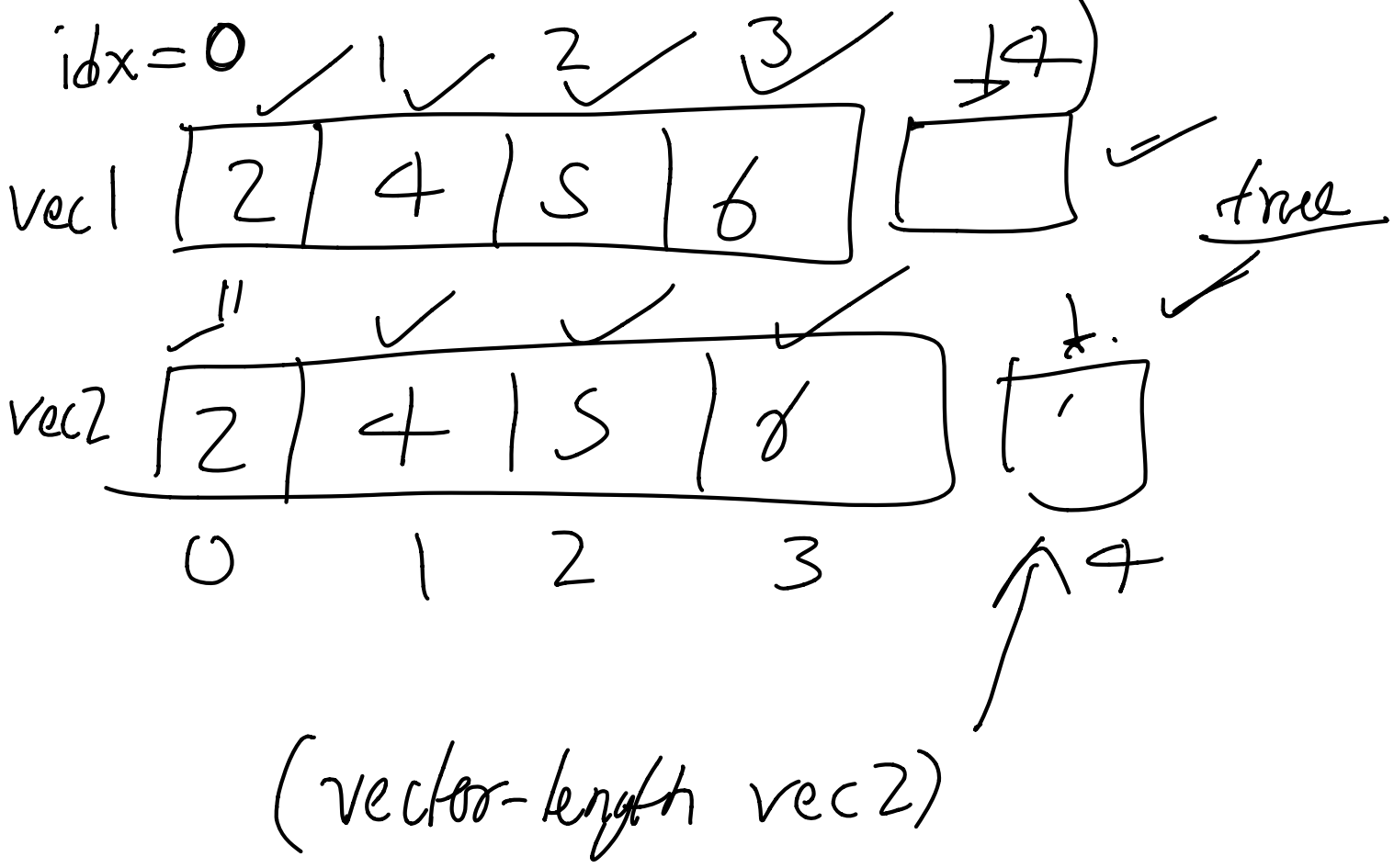
false

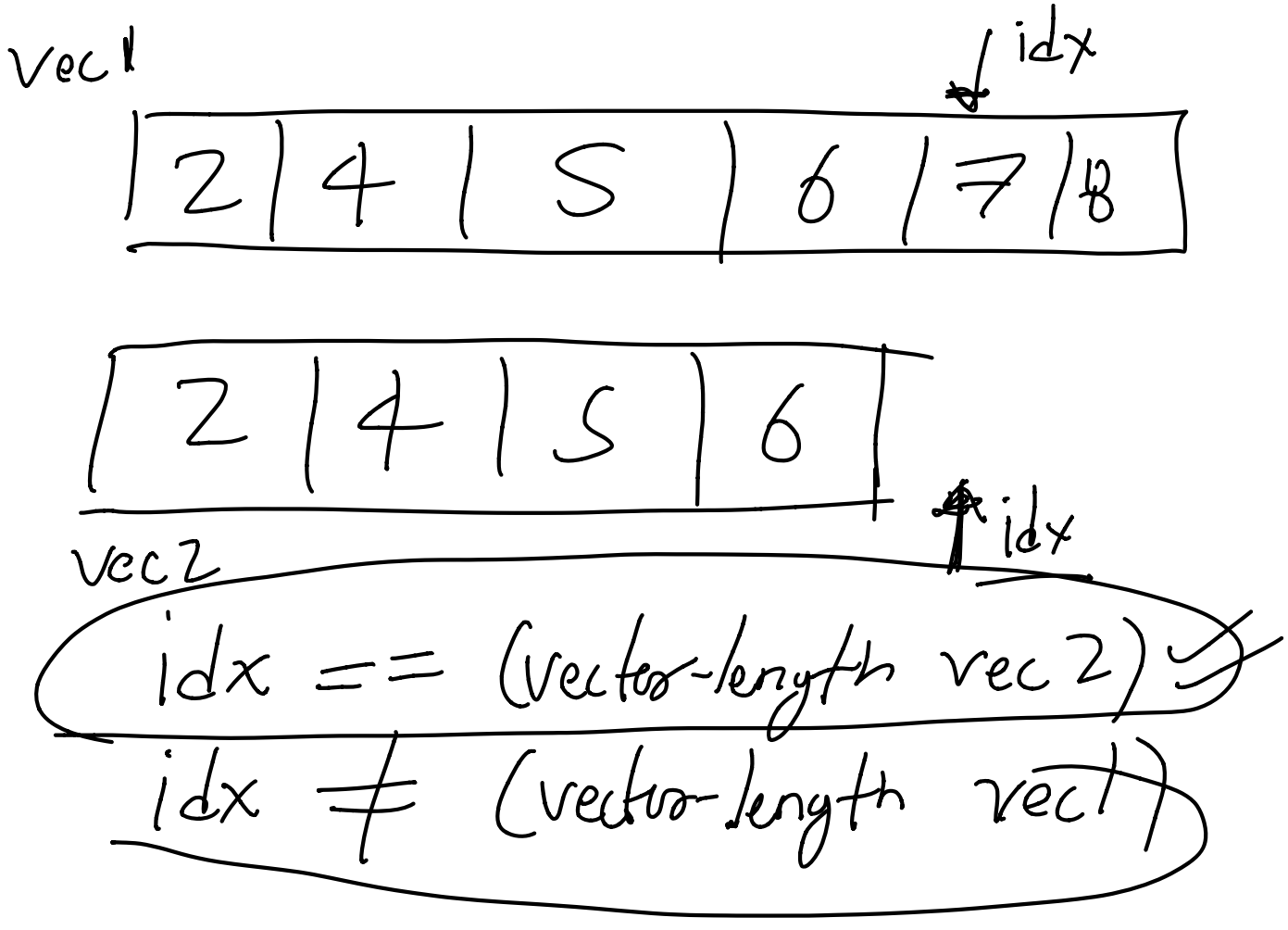
2nd

3rd

Start index

(vector-length vec1)


idx=0
1
2
3
4
vec1
2
4
5
6
free
vec2
2
4
5
6
0
1
2
3
4
(vector-length vec2)


vec1
idx
2
4
5
6
7
8
2
4
5
6
idx
vec2
idx == (vector-length vec2)
idx ≠ (vector-length vec1)

Check expect for HW10

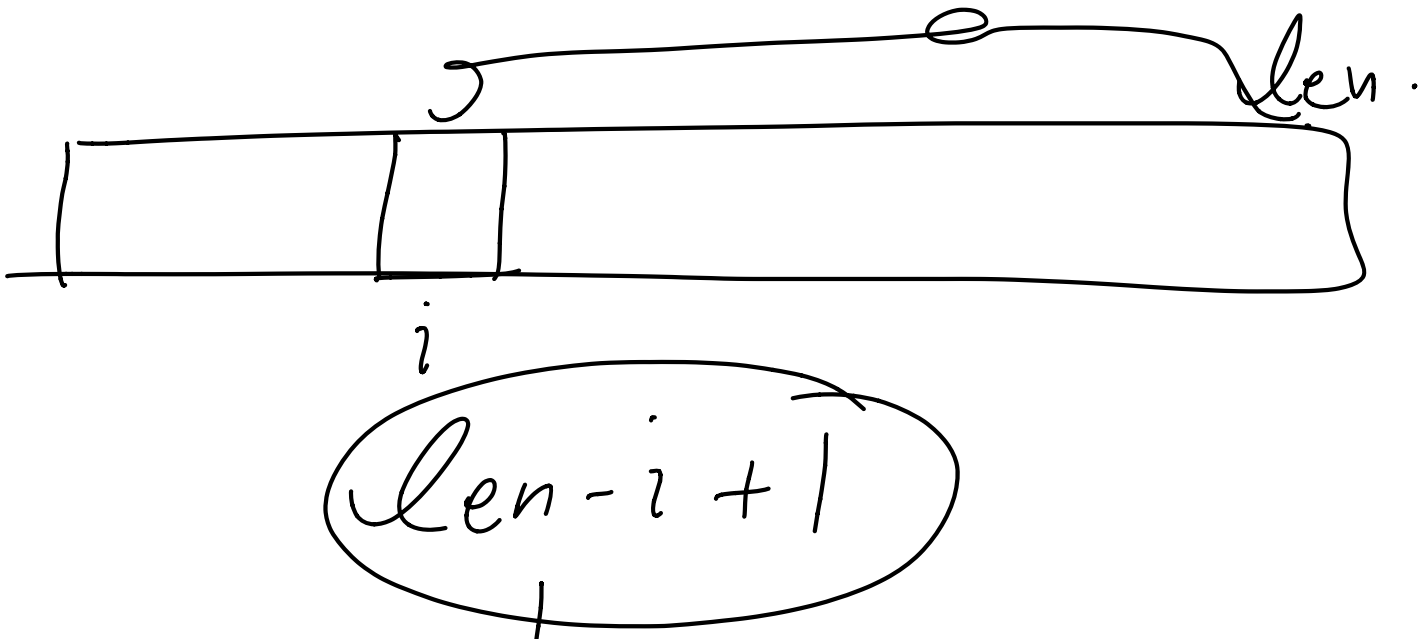
```
(define my-vec (vector 1 2 3 4))

(add-one! my-vec)

(check-expect my-vec (vector
                      2 3 4 5))

(test)
```

Recurrence for the helper function

$$T(n) = T(n-1) + \underbrace{O(1)}_{\text{Constant}}$$

What is n ?

n - Input size



Number of elements to be scanned

$n \triangleq$ (vector-length input-vector) - idx [Number of elements on the right side]

Recurrence for the helper function

is $T(n) = T(n-1) + C$

$T(n) = O(n)$

Time-complexity of add-one

$T($(vector-length input-vector)$)$

length of input-vector

$- O)$

$O($length of input-vector$)$

Final exam

Problem 3b

(Hemisphere- longitude) — Not allowed

(Hemisphere- latitude) — Not allowed

(match hem
[(Hemisphere lo la) [

```
(match* (hem1 hem2)
[((Hemisphere lat1 lo1) (Hemisphere lat2
                                  lo2)
```

```
(begin (vector-set! vec i
                    (+ (vector-ref vec i) 1))
       (add-one-helper! (vec (+ i 1))).
```

```
(begin (vector-set! vec i
                    (+ (vector-ref vec i) 1))
       (define j (+ i 2)))
```

(add-one helper
vec j))

+1