Prolegomena unit outline:

- Algorithms and correctness (Monday and **today**)
- Algorithms and efficiency (**today** and next week Wednesday and Friday)
- Abstract data types (the following Wednesday)
- Data Structures (the following Wednesday and Friday)

Today:

- Finish check-sorting problem
- "Binary search" problem
- Class invariants (`LinkedList`)
- Start efficiency

What good are invariants?

- They are a tool for reasoning about the state and progress of an algorithmic process
- They are a way to explain the meaning of a variable and capture how the variables relate to each other.
- They help with testing and debugging.
- They are a means for proving that an algorithm is correct.

Invariant (Class `LinkedList`)

(a) `head` $=$ `null` *iff* `tail` $=$ `null` *iff* `size` $= 0$.

(b) *If* `tail` $\neq$ `null` *then* `tail.next` $=$ `null`.

(c) *If* `head` $\neq$ `null` *then* *`tail`* *is reached by following* `size` $- 1$ *`next`* *links from* *`head`*.

```
def bounded_linear_search(sequence, P):
    found = False                                   # a0
    i = 0
    while not found and i < len(sequence):          # a1(n+1)
        found = P(sequence[i])                      # a2 n
        i += 1
    if found:                                       # a3
        return i - 1                                # a4
    else :
        return -1                                   # a5
```

$$\begin{aligned} T_{bls}(n) &= a_1 + a_2(n+1) + a_3 n + a_4 + \max(a_5, a_6) \\ &= b_0 + b_1 n \end{aligned}$$

```
def binary_search(sequence, TO, item):
    low = 0                                          # c0
    high = len(sequence)
    while high - low > 1 :                           # c1(lg n + 1)
        mid = (low + high) / 2                       # c2 lg n
        compar = TO(item, sequence[mid])
        if compar < 0 :     # item comes before mid
            high = mid
        elif compar > 0 :   # item comes after mid
            low = mid + 1
        else :              # item is at mid
            assert compar == 0
            low = mid
            high = mid + 1
    if low < high and TO(item, sequence[low]) == 0 : # c3
        return low                                   # c4
    else :
        return -1                                    # c5
```

$$\begin{aligned} T_{bs}(n) &= c_0 + c_1(\lg n + 1) + c_2 \lg n + c_3 + \max(c_4, c_5) \\ &= d_0 + d_1 \lg n \end{aligned}$$

```
def selection_sort(sequence, TO):
    for i in range(len(sequence)):                          e_0 + e_1 n
        min_pos = i
        min = sequence[i]
        for j in range(i + 1, len(sequence)):               e_3 n + e_4 Σ_{i=0}^{n-1} (n - i - 1)
            if TO(sequence[j], min) < 0 :                   e_2 n
                min = sequence[j]                           e_5 Σ_{i=0}^{n-1} (n - i - 1)
                min_pos = j
        sequence[min_pos] = sequence[i]
        sequence[i] = min
```

$$T_{sel}(n) = f_1 + f_2 n + f_3 n^2$$

Coming up:

By class time next week Wednesday:
Do Ex 1.(6 & 7)
Take quiz

By midnight next week Wednesday:
Read Section 1.2

For next week Friday:
Read Sections 1.(3 & 4)
Do practice problems 1.(27 & 28) and 1.(42 & 43)
Take quiz