def linear_search(A, v):
    i = 0
    while i < len(A) and A[i] != v :
        i = i + 1
    if i == len(A) :
        return None
    else
        return i
```python
def linear_search(A, v):
    i = 0
    while i < len(A) and A[i] != v:
        i = i + 1
    if i == len(A):
        return None
    else:
        return i
```

- \( \forall k \in [0, i), A[k] \neq v \).
- \( i \) is the number of iterations completed.

**Init.** Initially, \( i = 0 \), so both parts of the invariant are trivially true.

**Maint.** Suppose that before the iteration, \( \forall k \in [0, i), A[k] \neq v \), and \( i \) is the number of iterations so far.

In order for the iteration to be executed, \( A[i] \neq v \). The body of the loop implies \( i_{\text{post}} = i_{\text{pre}} + 1 \). Then \( \forall k \in [1, i_{\text{post}}), A[k] \neq v \).

Moreover, \( i_{\text{post}} \) is now the number of iterations so far.

(This completes the proof of the lemma that the proposition above is a loop invariant.)
```python
def linear_search(A, v):
    i = 0
    while i < len(A) and A[i] != v:
        i = i + 1
    if i == len(A):
        return None
    else:
        return i
```

Term. By the loop invariant, after \( n \) iterations \( i = n \) and so the guard fails after no more than \( n \) iterations. When the guard fails, either \( A[i] = v \) or \( i = n \). In either case, the loop terminates after at most \( n \) iterations.

In the first case, \( A[i] = v \), and \( i \) is returned. Moreover, by the loop invariant \( i \) is the first position in \( A \) that contains \( v \).

In the second case \( i = n \) and \( \text{None} \) is returned. By the loop invariant we know that \( \forall k \in [0, n), A[k] \neq v \) and so \( v \) exists nowhere in \( A \). Either way the algorithm is correct.
def linear_search(A, v):
    found = False
    i = 0
    while not found and i < len(A):
        found = A[i] = v
        i = i + 1
    if found:
        return i - 1
    else:
        return None

Invariant:

▶ $\forall k \in [0, i - 1), A[k] \neq v$.
▶ $\text{found iff } A[i - 1] = v$
▶ $i$ is the number of iterations completed
def selection_sort(A):
    for i in range(len(A)):
        min_pos = i
        min = A[i]
        for j in range(i + 1, len(A)):
            if A[j] < min:
                min = A[j]
                min_pos = j
        A[i] = min
For next time

*Read Section 2.3*

*Do Ex 2.3-(3, 6, 7)*

*See special instructions for 2.3-7*