Middlebury

CSCI 201: Data Structures

Spring 2025

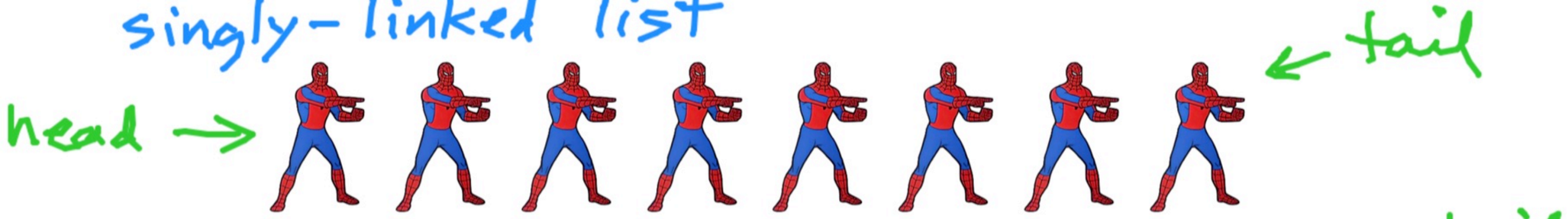
Lecture 6W: More Linked Lists

Goals for today:

- Implement a **doubly** linked list.
- Introduce the **circular** linked list.
- **Analyze** the performance of various linked list operations.
- **Generic**-ify our linked list implementation.

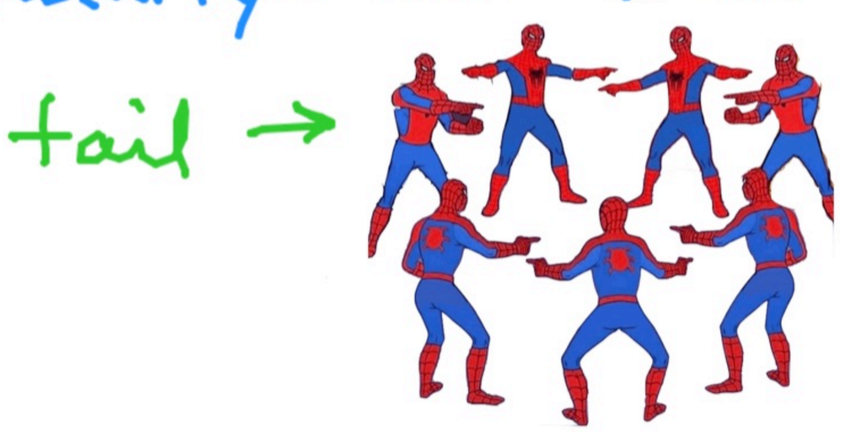
Types of linked lists

singly-linked list



doubly-linked list

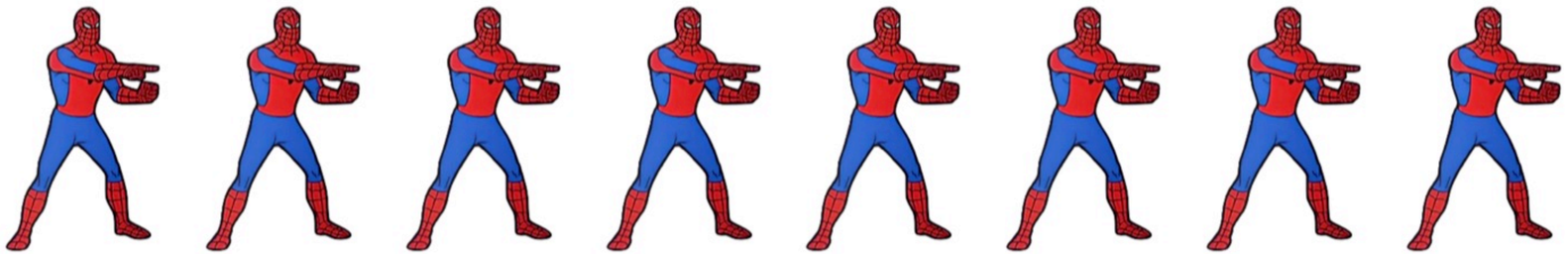
circularly-linked list



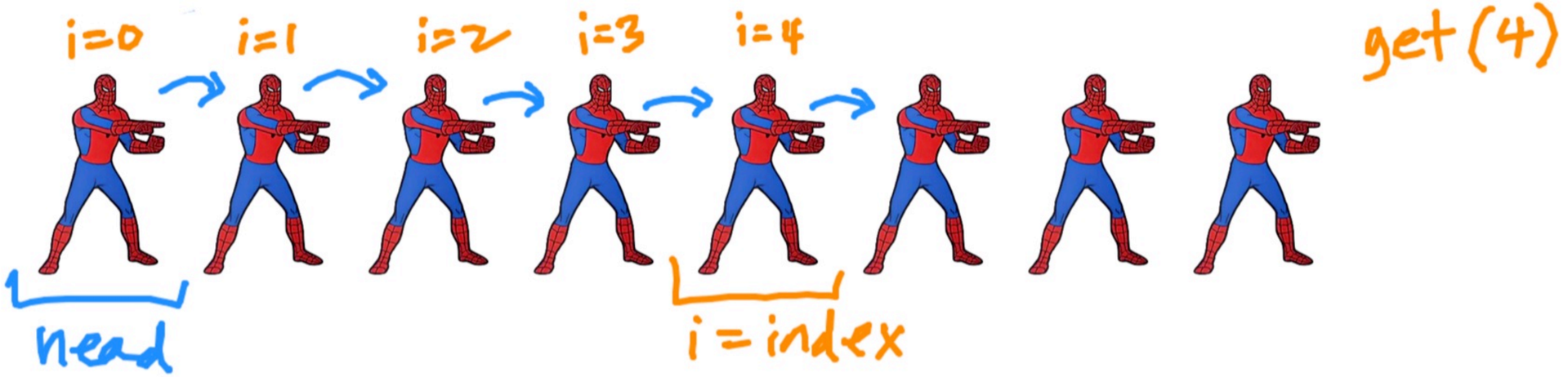
Redesigning our `LinkedList` (for any type)!

```java
class ListNode<E> {
  public ListNode<E> next;
  private E data;

  public ListNode(E data) {
    this.data = data;
    next = null;
  }

  public E get() {
    return data;
  }
}
```

```java
public class LinkedList<E> {
  private ListNode<E> head;

  public LinkedList() {
    head = null;
  }
}
```



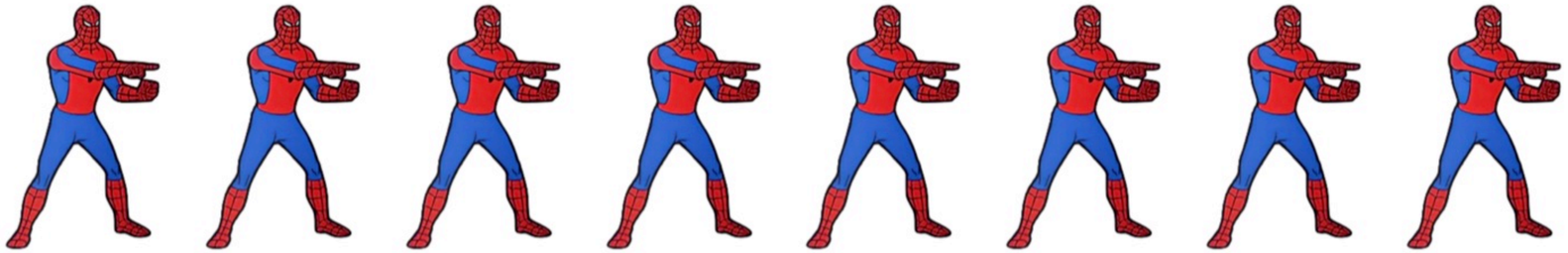
What if we want to get an item at a given index?



Start at head and follow next pointers

```java
public E get(int index) {
  ListNode<E> node = head;
  for (int i = 0; i < index; i++) {
    node = node.next;
  }
  return node.get();
}
```

Exercise 1: implement `size()` to determine # items in the list.



n items

O(n)

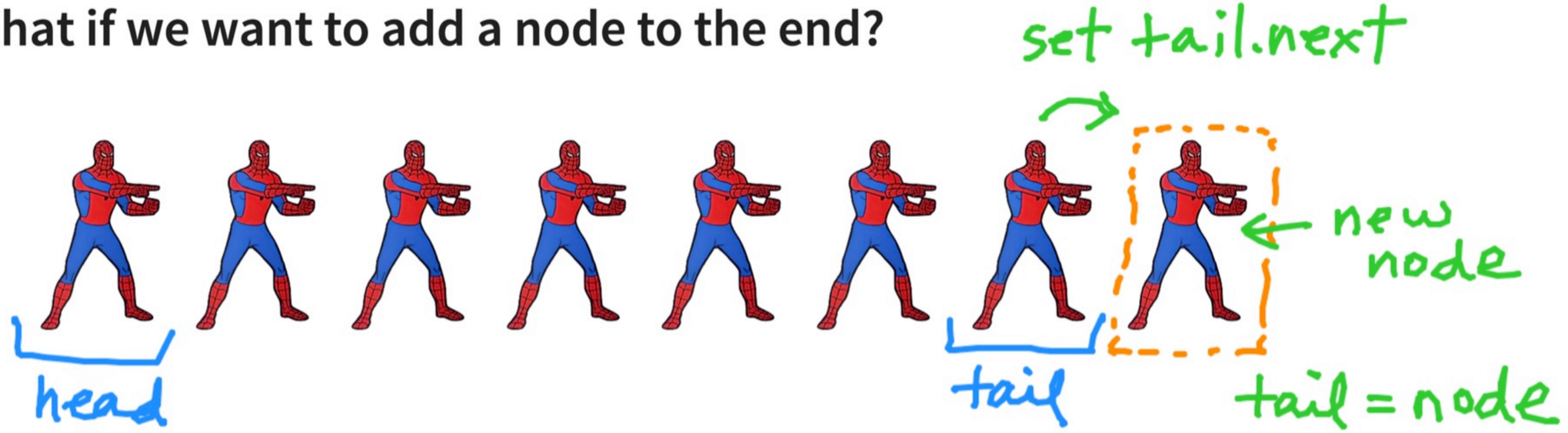
```java
int size() {
  ListNode<E> node = head;
  int nItems = 0;
  while (node != null) {
    node = node.next;
    nItems++;
  }
  return nItems;
}
```

```java
public class LinkedList<E> {
  ListNode<E> head;
  int nItems;

  int size() {
    // keep track of nItems
    // with each add/remove
    return nItems;
  }
}
```

O(1)

but adds work to add & remove

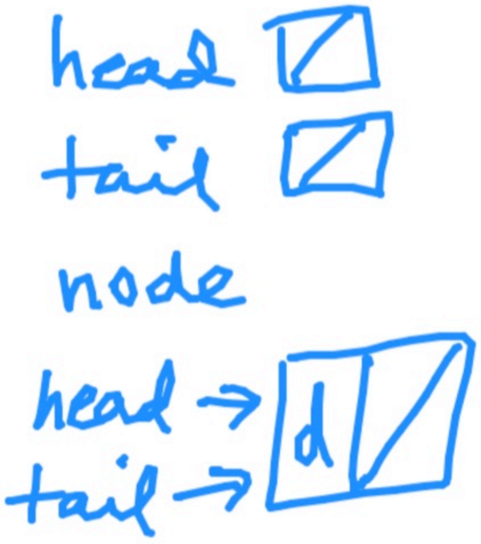
What if we want to add a node to the end?



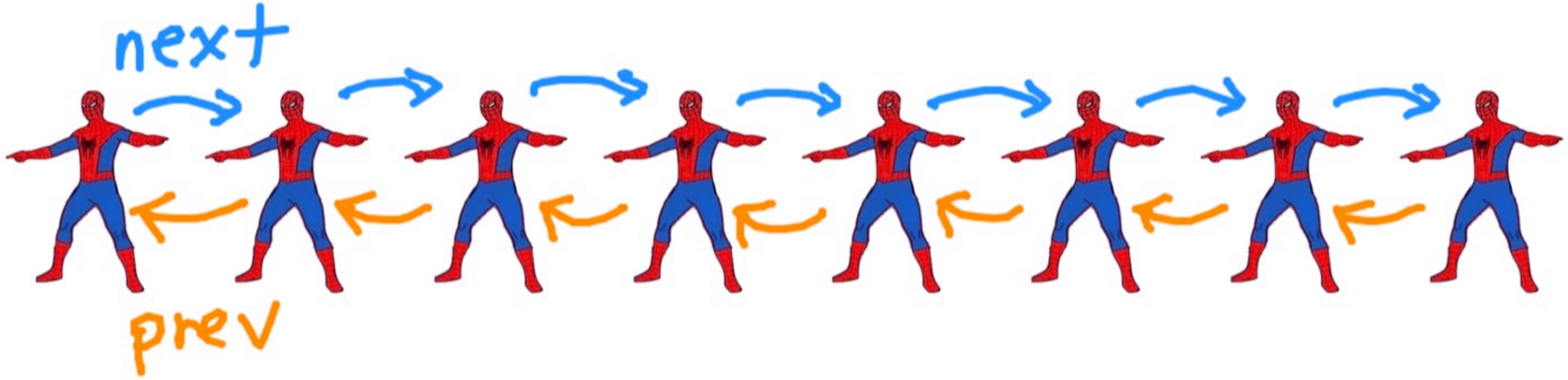
if (tail != null)

```
1  public class LinkedList<E> {
2    ListNode<E> head;
3    ListNode<E> tail;
4
5    void addLast(E data) {
6      // should check if head/tail is not null
7      ListNode<E> node = new ListNode<E>(data);
8      tail.next = node;
9      tail = node;
10   }
11 }
```

else head = node



And now let's add a `prev` field to each `ListNode`.
Why? What if we want to traverse the nodes backwards?




```java
class ListNode<E> {
  public ListNode<E> next;
  public ListNode<E> prev;
  private E data;

  public ListNode(E data) {
    this.data = data;
    next = null;
    prev = null;
  }

  public E get() {
    return data;
  }
}
```

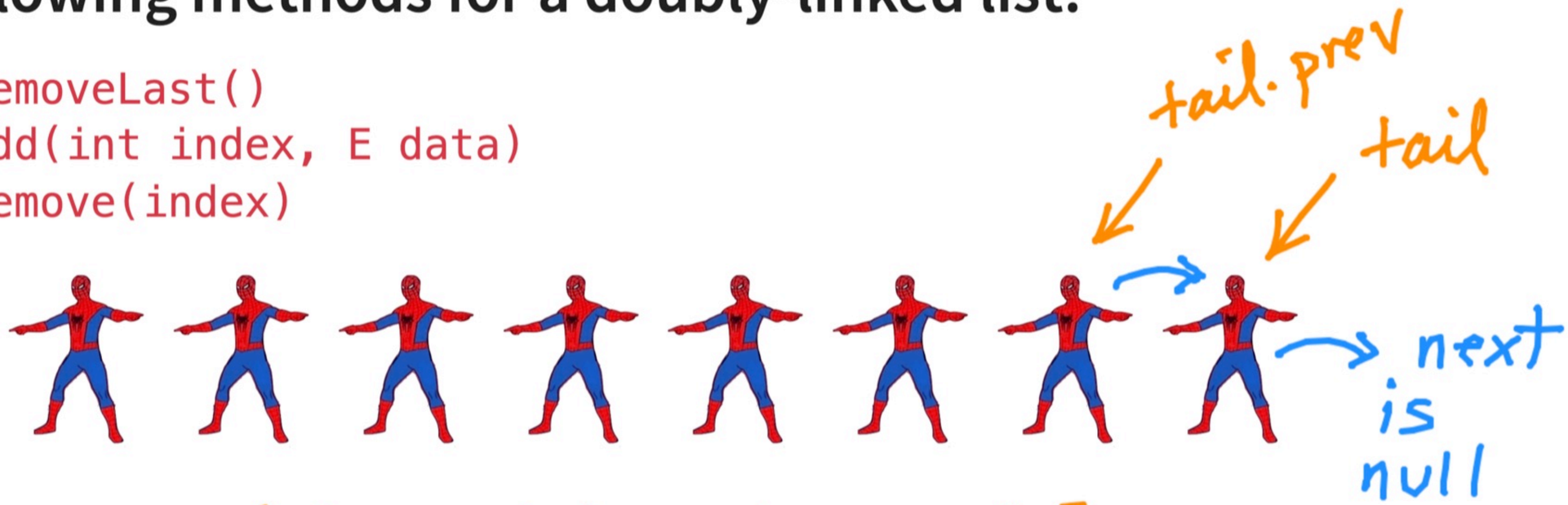
How do `addFirst` and `addLast` change with a doubly-linked list?

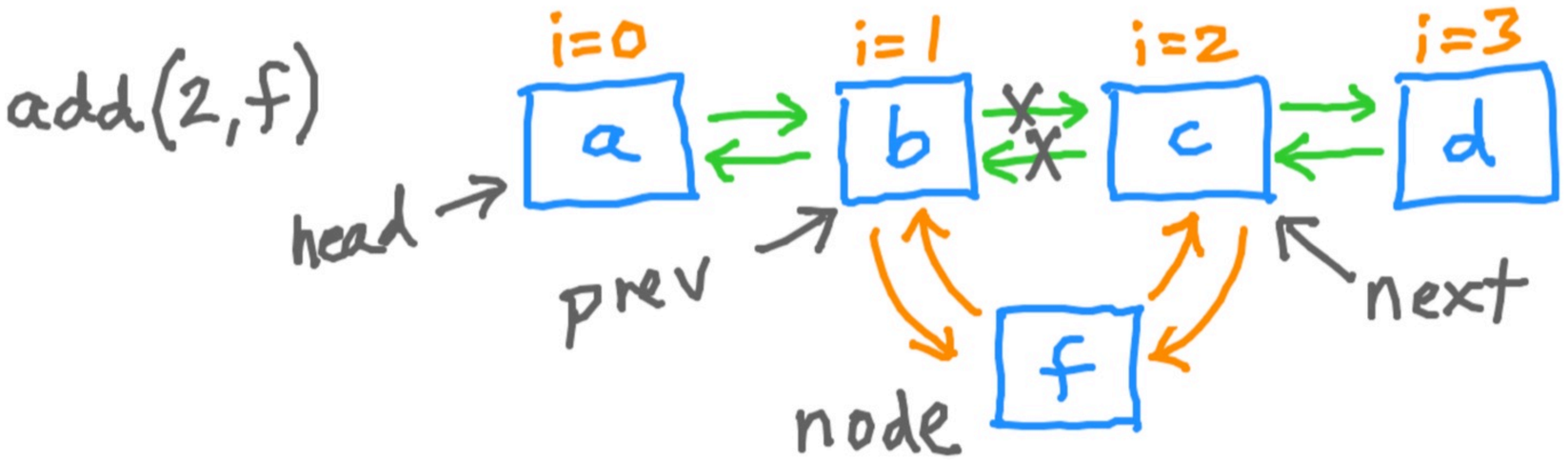
```java
public void addFirst(E data) {
  ListNode<E> node = new ListNode<E>(data);
  node.next = head;
  if (head != null) {
    head.prev = node;
  } else {
    // this is the first item so the tail is also the node
    tail = node;
  }
  head = node;
}
```

```java
public void addLast(E data) {
  // should check if head/tail is null
  ListNode<E> node = new ListNode<E>(data);
  node.prev = tail;
  tail.next = node;
  tail = node;
}
```

Exercises: use a LiveShare server to complete the following methods for a doubly-linked list.

- `removeLast()`
- `add(int index, E data)`
- `remove(index)`





See you Friday!

- Homework 5 due tomorrow (3/27) at 11:59pm.
  - Implement a variant of `Merge Sort`.
- Lab 5 Friday will involve writing our own text editor:
  - Think about how you would represent a line of characters using a linked list, and how to represent the cursor.
- Midterm Study Guide is posted on Canvas.
- Reminder that Noah (go/noah) and Smith (go/smith) have office hours throughout the week and the 201 Course Assistants have drop-in hours in the late afternoons/evenings (go/cshelp).