LEC 12

CSE 123

Linked Lists with Recursion

Questions during Class?

Raise hand or send here

sli.do #cse123A

BEFORE WE START

Talk to your neighbors:

Best boba in Seattle?

Instructors: Nathan Brunelle

TAs:	Arohan	Ashar	Neha	Rohini	Rushil
	Ido	Zachary	Sebastian	Joshua	Sean
	Hayden	Caleb	Justin	Heon	Rashad
	Srihari	Benoit	Derek	Chris	Bhaumik
	Kuhu	Kavya	Cynthia	Shreya	Ashley
	Ziao	Kieran	Marcus	Crystal	Eeshani
	Prakshi	Packard	Cora	Dixon	Nichole
	Niyati	Trien	Lawrence	Evan	Cady

Lecture Outline

- Announcements
- Traversing Linked Lists Recursively
- Modifying Linked Lists Recursively

Announcements

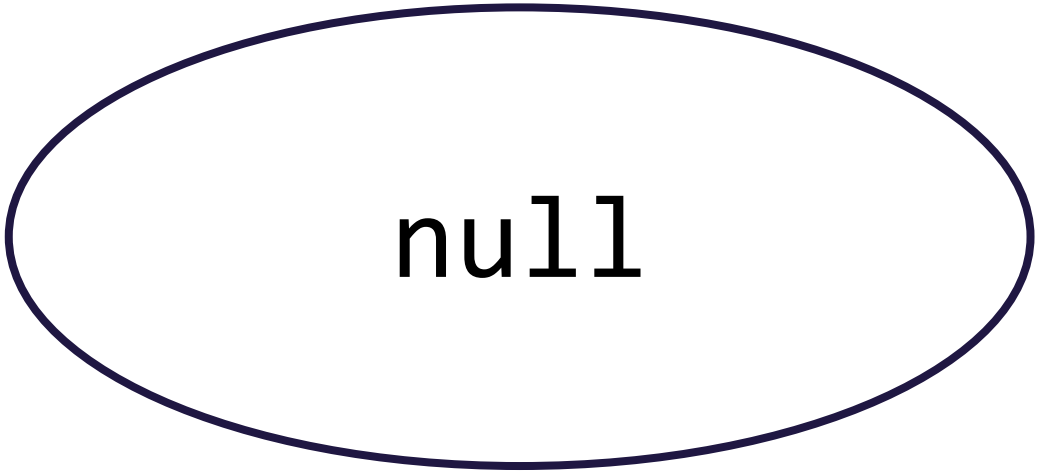
- Creative Project 2 due tonight May 14 at 11:59pm
- Resubmission Cycle 4 is due Fri (May 16) at 11:59pm
  - ***C1***, P1 eligible
- Programming Assignment 2 released tomorrow (May 15)
  - Focused on exhaustive search + recursive backtracking!
- Quiz 1 grades out early next week

Lecture Outline

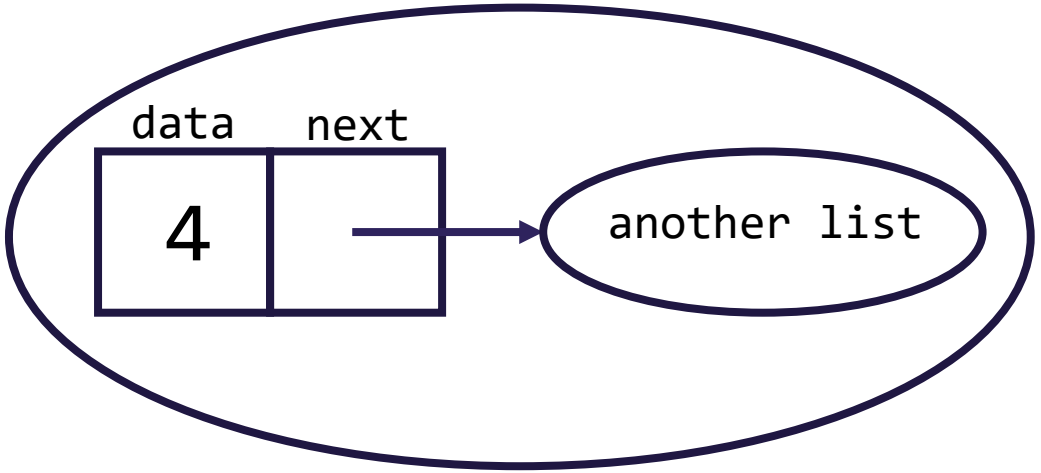
- Announcements
- **Traversing Linked Lists Recursively**
- Modifying Linked Lists Recursively

Linked Lists

- A linked list is either:



Empty list



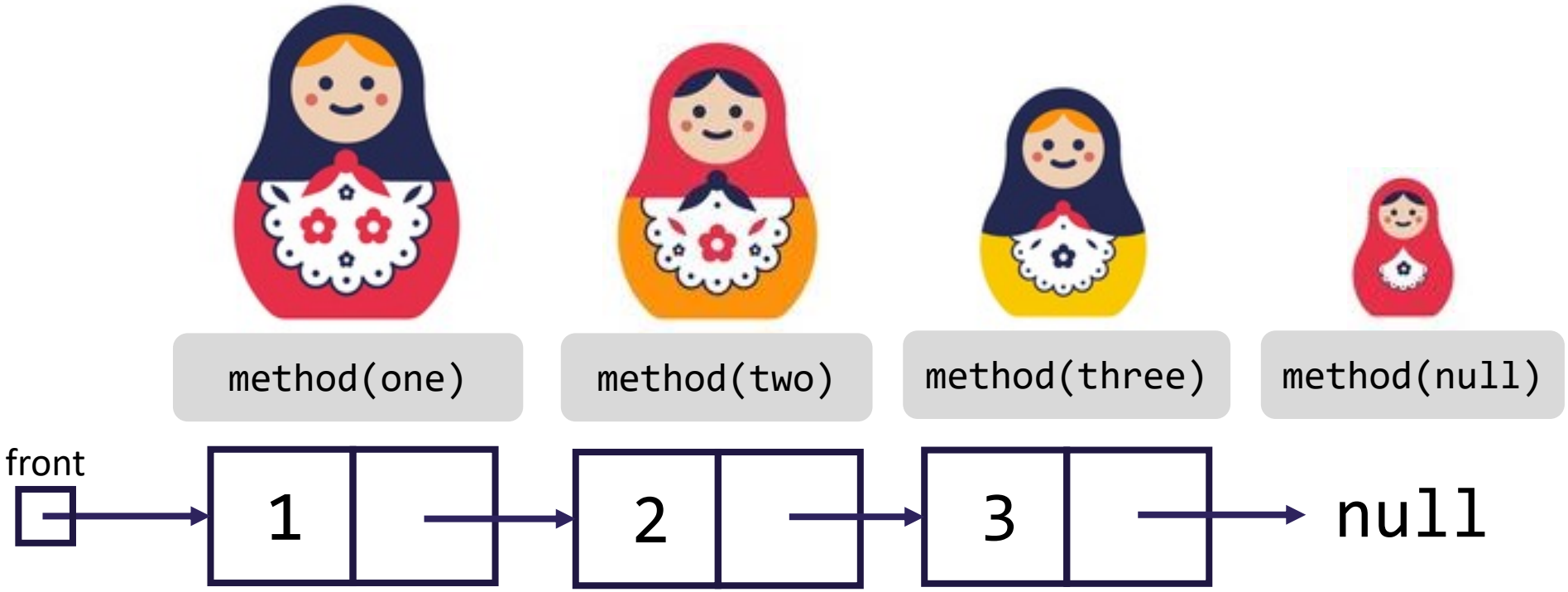
Node w/ another linked list

This is a recursive definition!

A list is either empty or a node with another list!

Recursive Traversals w/ LinkedLists

- Guaranteed base case: empty list
  - Simplest possible input, should immediately know the return
- Guaranteed public / private pair
  - Need to know which sublist you're currently processing (i.e. `curr`)



Lecture Outline

- Announcements
- Traversing Linked Lists Recursively
- **Modifying Linked Lists Recursively**

Modifying LinkedLists [Review]

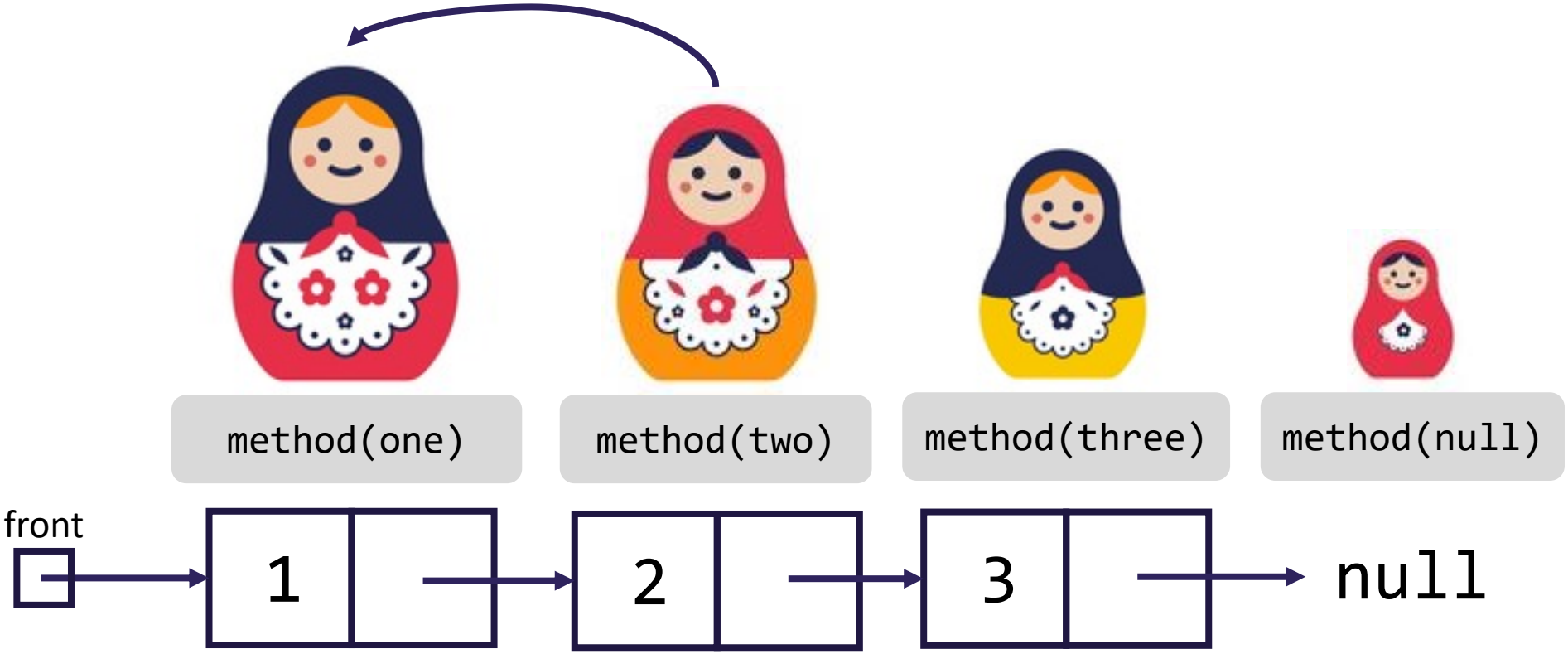
- Remember: using a `curr` variable to iterate over nodes

- Does changing `curr` actually update our chain?
  - What will? Changing `curr.next`, changing `front`
  - Need to **stop one early** to make changes

- Often a number of cases to watch out for:
  - M(iddle) – Modifying node in the middle of the list (general)
  - F(ront) – Modifying the first node
  - E(mpty) – What if the list is empty?
  - E(nd) – Rare, do we need to do something with the end of the list?

Modifying LinkedLists Recursively

- Much easier than iterative solutions!
- No longer need to stop one early
  - Can go right to the point you'd like to make the change



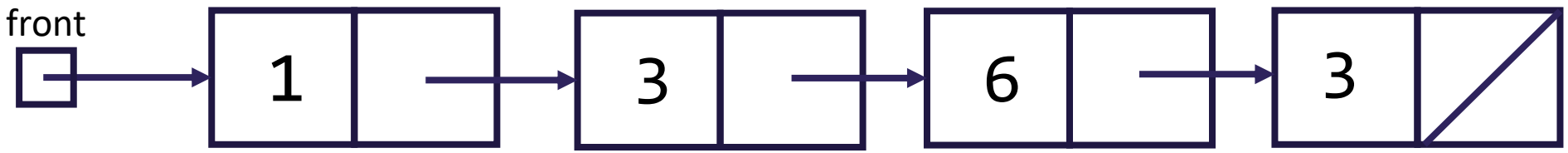
Modifying LinkedLists Recursively

- Much easier than iterative solutions!
- No longer need to stop one early
  - Can go right to the point you'd like to make the change

- How? Return the updated change and catch it!
  - Private pair returns `ListNode` type
  - `curr.next = change(curr.next)` / `front = change(front)`
  - Resulting solutions much cleaner than iterative cases

- We call this pattern **`x = change(x)`**

removeAll Walkthrough




```
private ListNode removeAll(int value, ListNode node) {
    if (node == null) {
        return node;
    } else if (node.data == value) {
        return removeAll(value, node.next);
    } else {
        // x = change(x)
        node.next = removeAll(value, node.next);
        return node;
    }
}
```