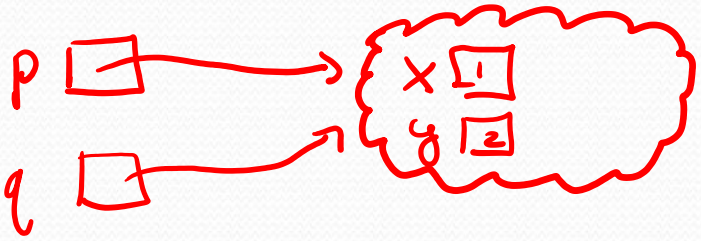
# Warm Up: pollev.com/cse143

```
Point p = new Point(1,2);
Point q = p;
```



# Road Map

## CS Concepts

- Client/Implementer
- Efficiency
- Recursion
- Regular Expressions
- Grammars
- Sorting
- Backtracking
- Hashing
- Huffman Compression

## Java Language

- Exceptions
- Interfaces
- References
- Comparable
- Generics
- Inheritance/Polymorphism
- Abstract Classes

## Data Structures

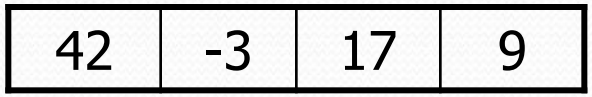
- Lists
- Stacks
- Queues
- Sets
- Maps
- Priority Queues

## Java Collections

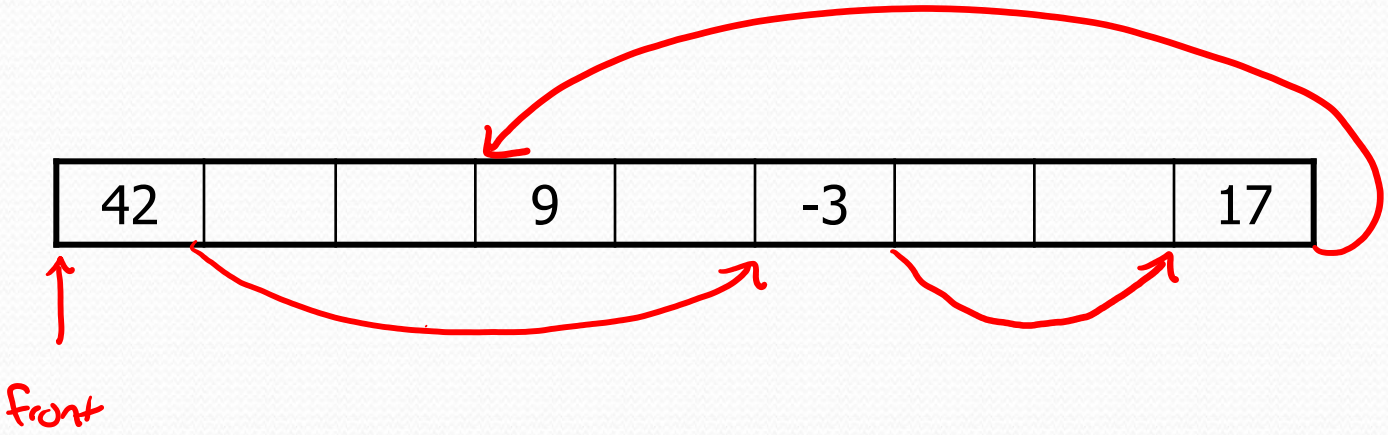
- Arrays
- ArrayList 
- LinkedList 
- Stack
- TreeSet / TreeMap
- HashSet / HashMap
- PriorityQueue

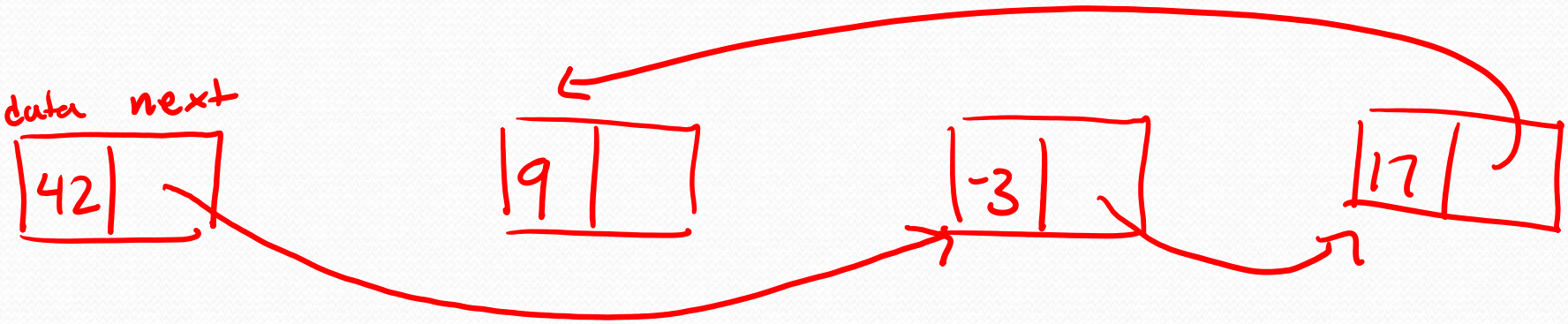
# Memory for a List

- Array (contiguous in memory)




- Spread in memory






pseudo code

ListNode

data (int)

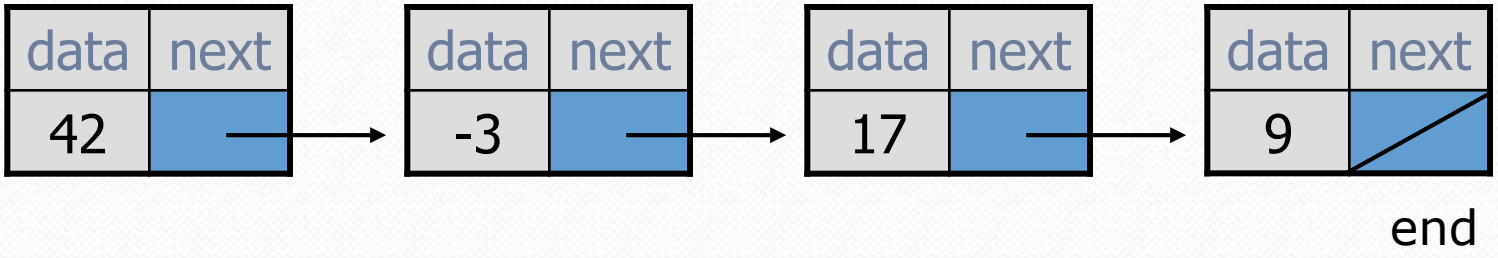
next (ListNode)

# A list node class

```
public class ListNode {
    public int data;
    public ListNode next;
}
```

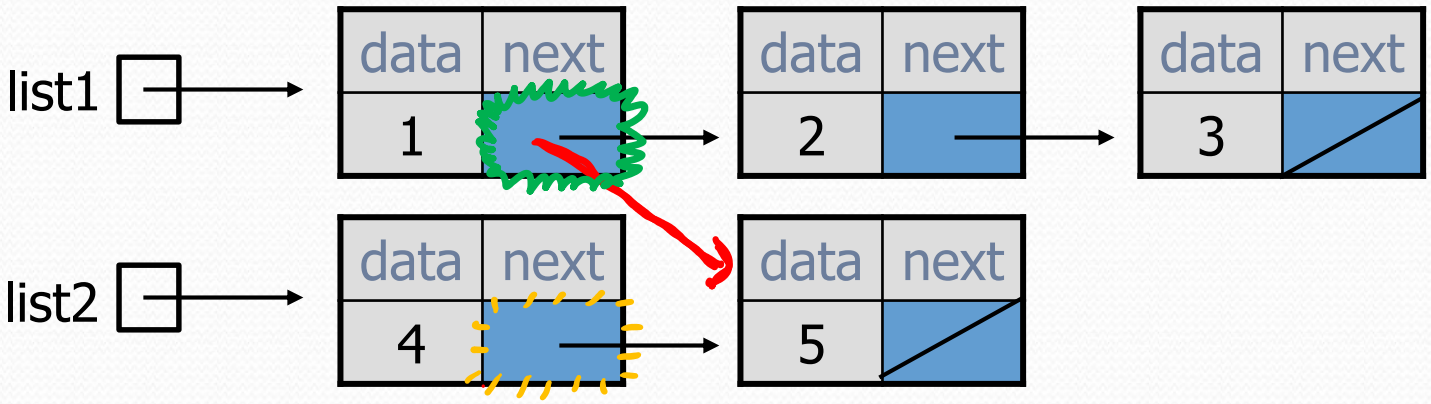
- Each list node object stores:
  - one piece of integer data
  - a reference to another list node

- `ListNode`s can be "linked" into chains to store a list of values:



Poll Everywhere pollev.com/cse143

- Suppose we had the following `ListNodes`:



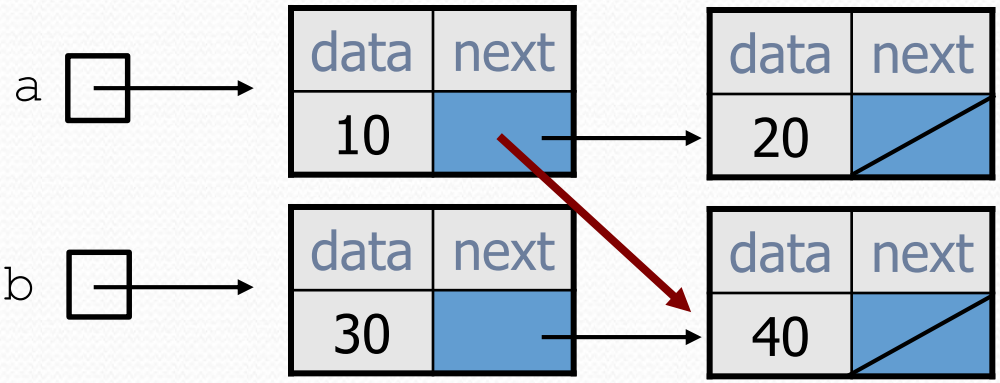

- What would the lists look like if we ran the code?

```
list1.next = list2.next;
```

# Reassigning references

- when you say:
  - `a.next = b.next;`

- you are saying:
  - "Make *variable* `a.next` store to the same *value* as `b.next`."
  - Or, "Make `a.next` refer to the same place as `b.next`."

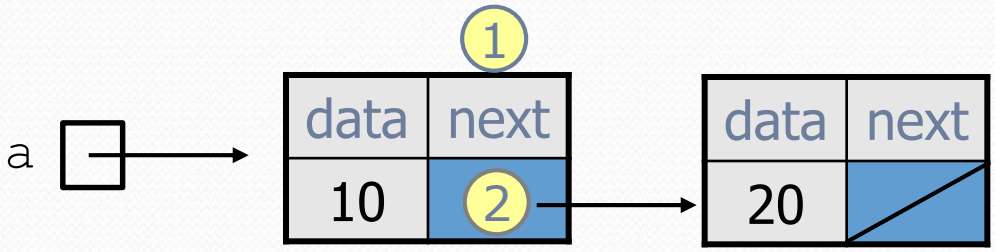


# References vs. objects

**variable** = **value**;

a *variable* (left side of = ) place to put a reference
(where the phone number goes; where the base of the arrow goes)
a *value* (right side of = ) is the reference itself
(the phone number; the destination of the arrow)

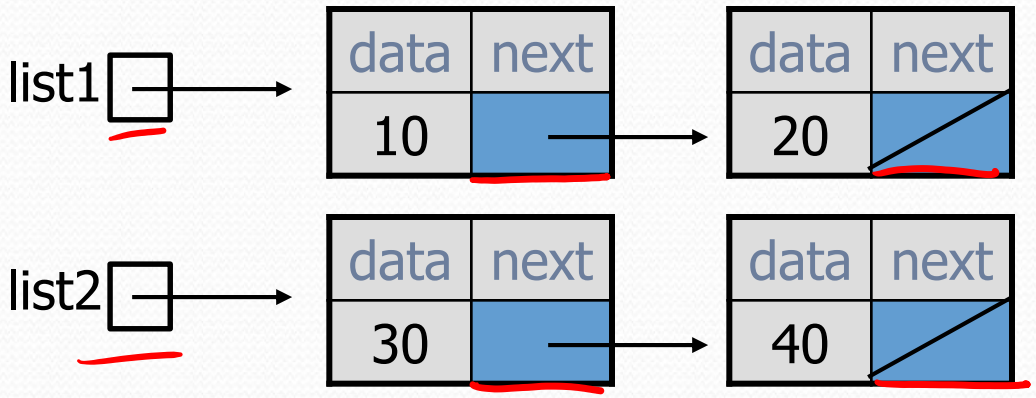
- adjus



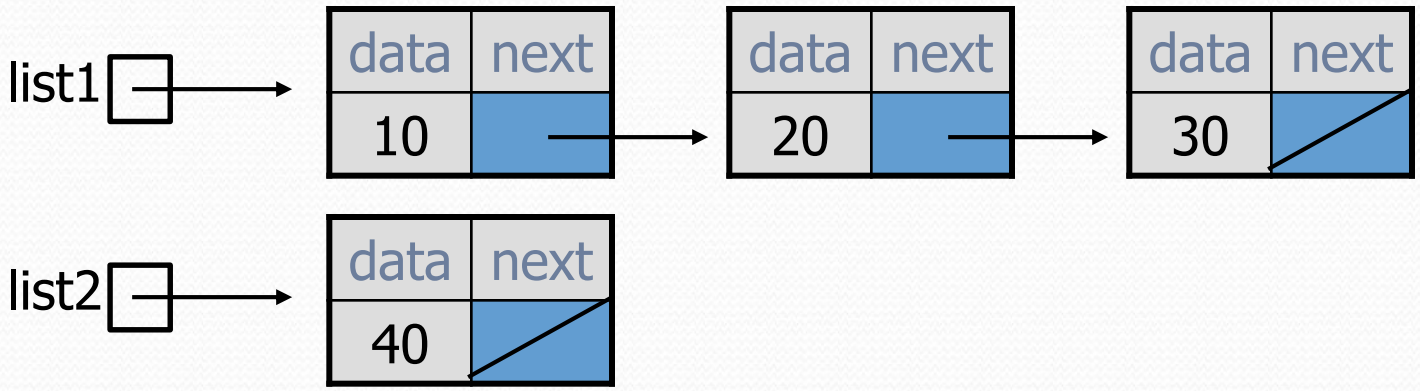

- For the list at right:
  - `a.next` = **value**; 1
    means to t where points
  - **variable** = `a.next`; 2
    means to make **variable** point at

# Linked node problem 3

- What set of statements turns this picture:



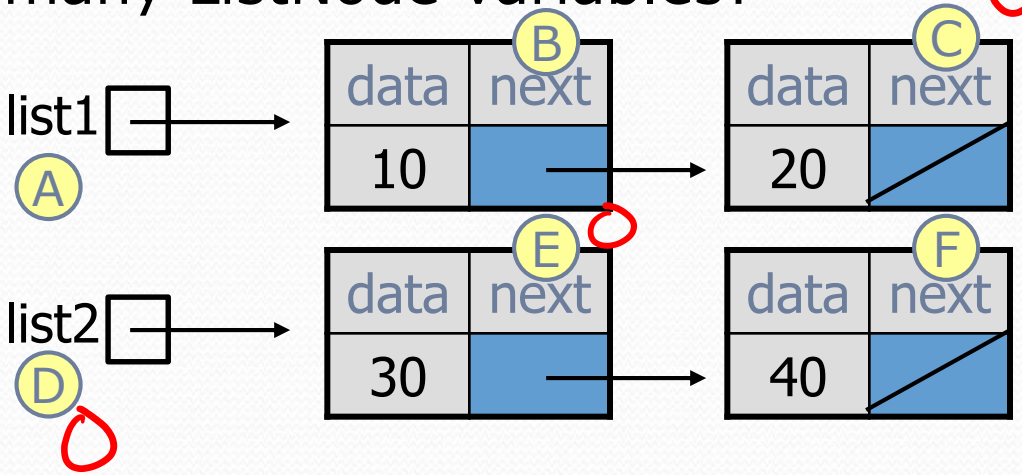

- Into this?



# Linked node problem 3

3 variables need to change!

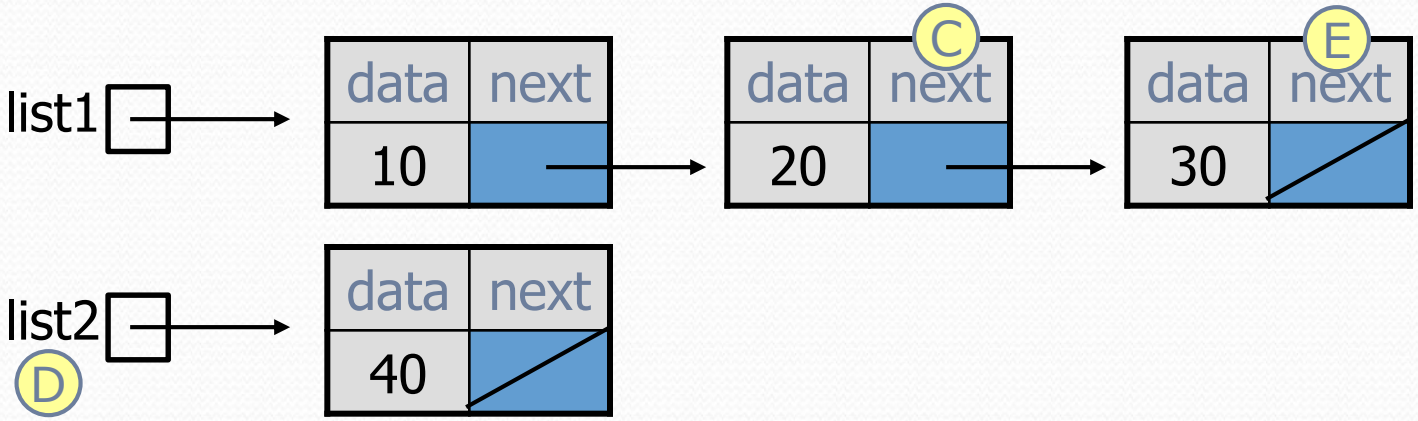
- How many ListNode variables?




C = D;

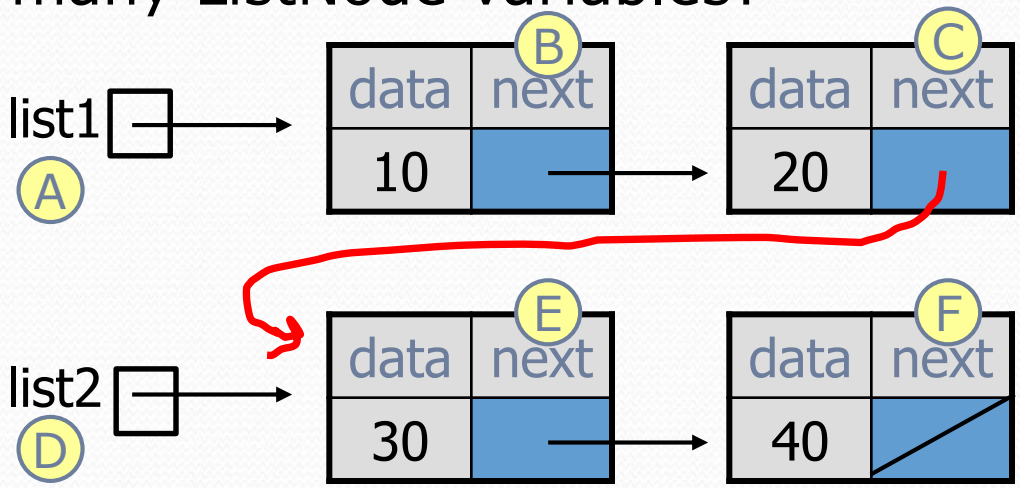
list1.next.next = 13+2;

- Which variables change?



# Linked node problem 3

- How many ListNode variables?




**list1.next.next = list2**

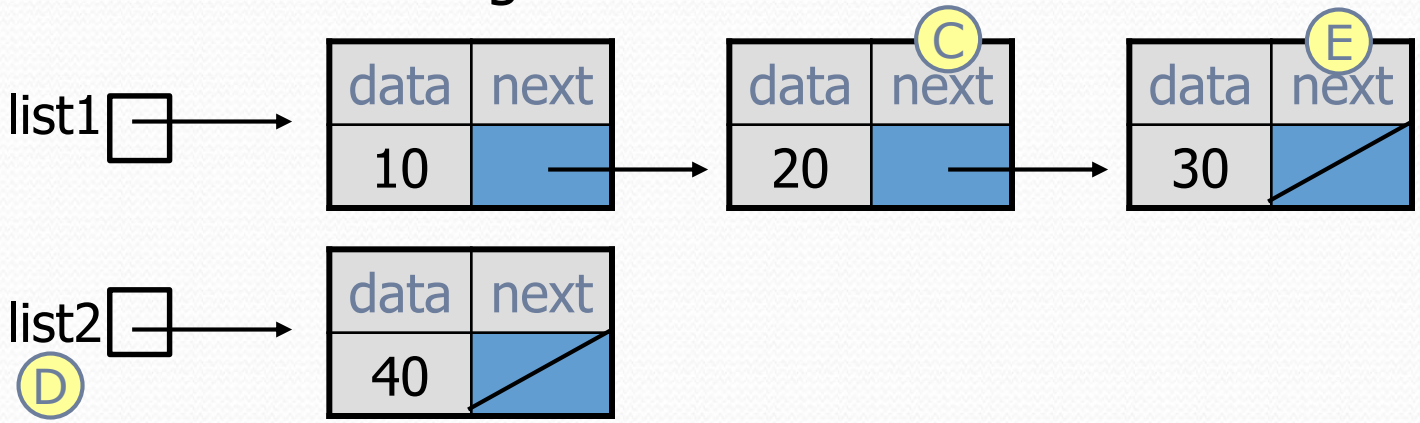
D = E;

l3+2 = l3+2.next;

or

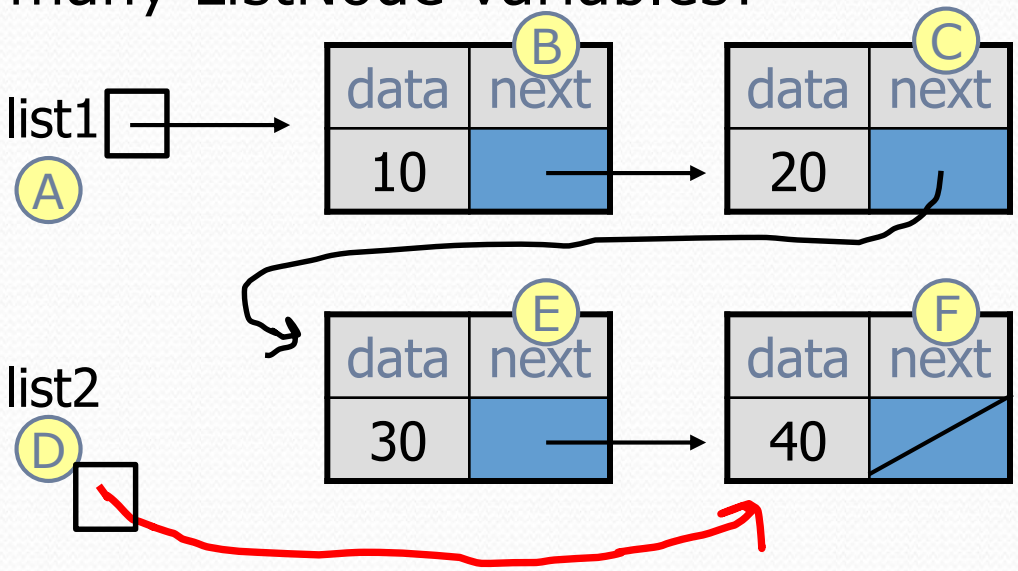
l3+2 = l3+1.next.next.next;

- Which variables change?



# Linked node problem 3

- How many ListNode variables?



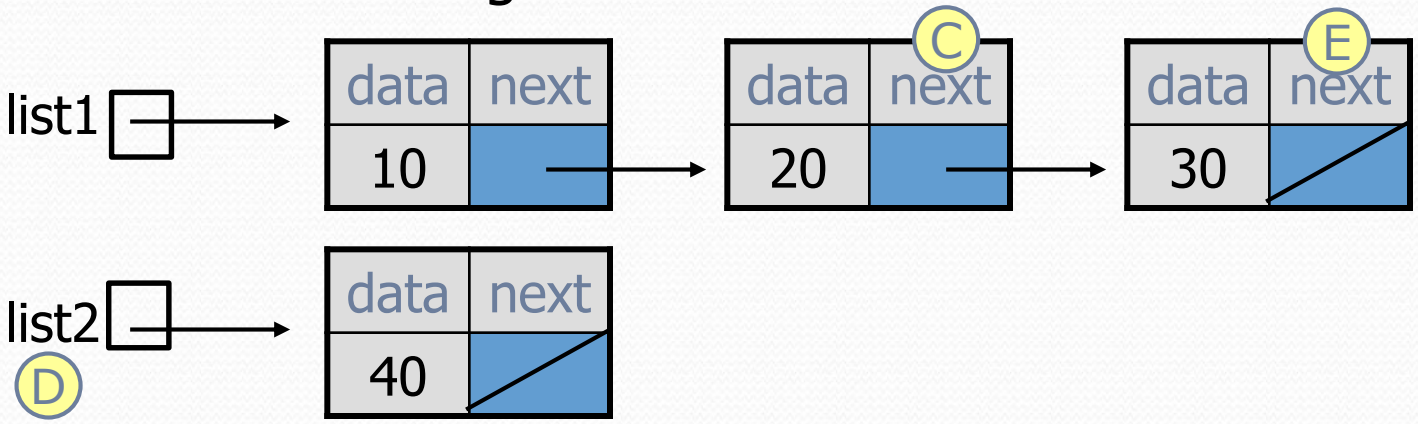

list1.next.next = list2

**list2 = list2.next**

E = null;

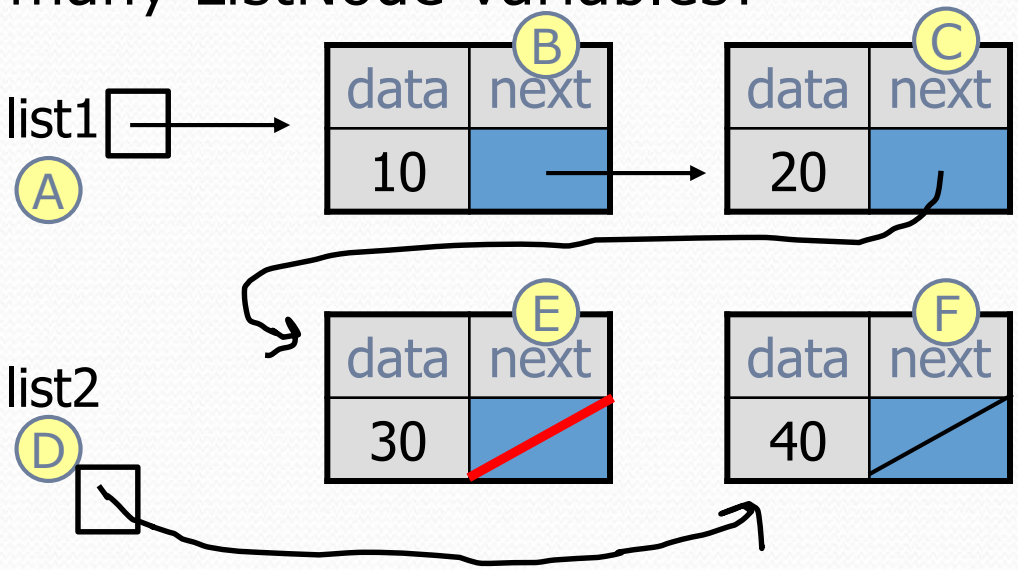
list1.next.next.next = null;

- Which variables change?



# Linked node problem 3

- How many ListNode variables?

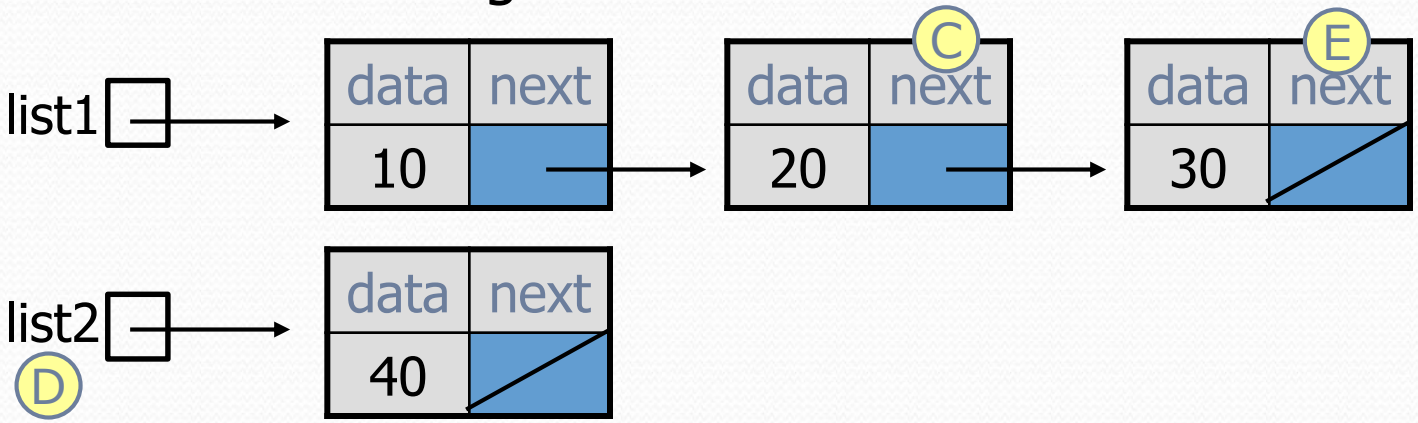



list1.next.next = list2

list2 = list2.next

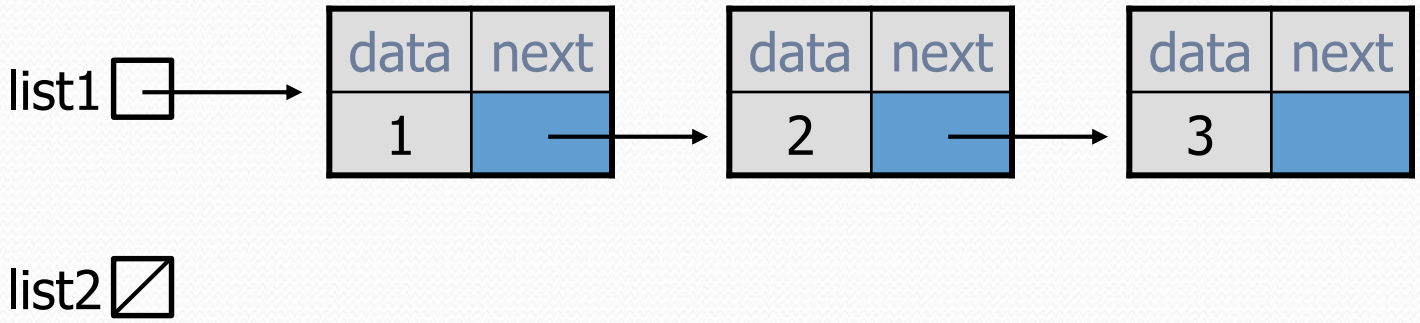
**list1.next.next.next = null**

- Which variables change?



# Linked node problem 4

- What set of statements turns this picture:




- Into this?