Linked node problem 3

- What set of statements turns this picture:

  list1  →  
  | data | next |
  | 10   |     |

  list2  →  
  | data | next |
  | 30   |     |

- Into this?

  list1  →  
  | data | next |
  | 10   |     |

  list2  →  
  | data | next |
  | 40   |     |

  | data | next |
  | 30   |     |
Linked node problem 3

• How many ListNode variables?

list1

A

list2

D

• Which variables change?
Abstract data types (ADTs)

- **abstract data type (ADT):** A specification of a collection of data and the operations that can be performed on it.
  - Describes *what* a collection does, not *how* it does it

- Java's collection framework describes several ADTs:
  - Queue, List, Collection, Deque, List, Map, Set

- An ADT can be implemented in multiple ways:
  - ArrayList and LinkedList implement List
  - HashSet and TreeSet implement Set
  - LinkedList, ArrayDeque, etc. implement Queue

- The **same** external behavior can be implemented in many different ways, each with pros and cons.
Linked List vs. Array

- Print list values:

```java
ListNode list = ...;

ListNode current = list;
while (current != null) {
    System.out.println(current.data);
    current = current.next;
}
```

- Similar to array code:

```java
int[] a = ...;

int i = 0;
while (i < a.length) {
    System.out.println(a[i]);
    i++;
}
```

<table>
<thead>
<tr>
<th>Description</th>
<th>Array Code</th>
<th>Linked List Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to front of list</td>
<td>int i = 0;</td>
<td>ListNode current = list;</td>
</tr>
<tr>
<td>Test for more elements</td>
<td>i &lt; size</td>
<td>current != null</td>
</tr>
<tr>
<td>Current value</td>
<td>elementData[i]</td>
<td>current.data</td>
</tr>
<tr>
<td>Go to next element</td>
<td>i++;</td>
<td>current = current.next;</td>
</tr>
</tbody>
</table>
Before/After

- Before

  ![](before.png)

- After

  ![](after.png)
changing a list

- There are only two ways to change a linked list:
  - Change the value of \text{front} (modify the front of the list)
  - Change the value of \text{<node>\.next} (modify middle or end of list to point somewhere else)

- Implications:
  - To add in the middle, need a reference to the \text{previous} node
  - Front is often a special case