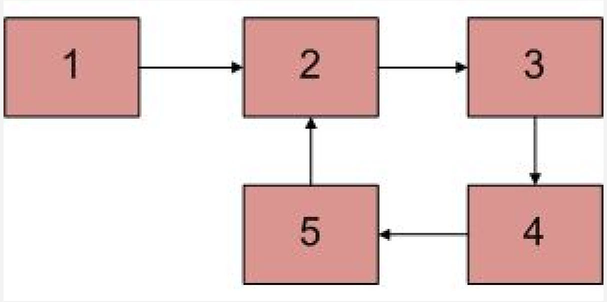
Technique: Slow Pointer, Fast Pointer

Level: Medium

Given a Linked List with a cycle, find the node where the cycle begins.

For example, node 2 in the below image:



Questions to Clarify:
Q. Can the entire list be a cycle?
A. Yes

Solution:
As done in the lecture video, detect a cycle, then find the length of the cycle.

Once you know the length of the cycle, it makes things easier. Let's say the length is L.

Simply take 2 pointers *A* and *B*, both at the start of the list. Move *A* forward by the L nodes.
Now, advance both pointers by 1 until they meet. The node at which they meet will be the start of the cycle.

Why does this work? Once you've found the length of the cycle (N nodes), then if 2 pointers are N nodes apart, they will eventually meet at the start when you move them forward together.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a few
lines quickly to plan out your solution. Your actual solution should be in a real
language and use good syntax.)

1. Advance fast and slow pointers until they meet
2. Find the length of the cycle, by advancing one of the
pointers around the cycle (as done in lecture solution)

3. Once you know the length L, move both pointers to head of list
advance one of the pointers by L

4. Now, keep advancing both pointers by 1 node, until they meet.
```

The node they meet is the starting point of the cycle.

Test Cases:
Edge Cases: no node in list
Base Cases: single element cycle
Regular Cases: cycle starts in middle, entire list is a cycle

Time Complexity: O(n)

Space Complexity: O(1)

```
public static Node findCycleStart(Node head) {
    // advance fast and slow pointers to meet in the cycle
    Node fast = head, slow = head;
    while (fast != null) {
        fast = fast.getNext();
        if (fast == slow)
            break;
        if (fast != null) {
            fast = fast.getNext();
            if (fast == slow)
                break;
        }
        slow = slow.getNext();
    }

    // no cycle found
    if (fast == null)
        return null;

    // find number of nodes in the cycle
    fast = fast.getNext();
    int cycleNodes = 1;
    while (fast != slow) {
        fast = fast.getNext();
        cycleNodes += 1;
    }

    // find start of cycle
    Node front = head, back = head;
    for(int i = 0; i < cycleNodes; i++) {
        front = front.getNext();
    }

    while (front != back) {
        front = front.getNext();
        back = back.getNext();
    }

    return front;
}
```