Technique: Delete Node

Level: Medium

Given a linked list and pointers to a node N and its previous node Prev, delete N from the linked list.

Questions to Clarify:
Q. Is N guaranteed to be in the list?
A. Yes

Q. What is Prev if N is the head?
A. Prev will be null

Solution:
If the node is head, we assign a new head. If it is tail, we assign the Previous node as the tail. Then, we assign Prev's next to N's next.

In languages with garbage collection (e.g, Java), N will be automatically deleted once there is no pointer to it. So we don't need to explicitly call delete.
Without garbage collection (like in C++), you also need to call 'delete' on N.

Note: If Prev was not given, we would need to iterate through the linked list to find N's previous node. That would take O(n) time.

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.)

delete(head, tail, N, Prev)
 if N is head, head = N.next
 if N is tail, tail = Prev
 prev.next = N.next
```

Test Cases:
Edge Cases: List empty, null values
Base Cases: Single element list, 2 element list
Regular Cases: N is head, tail, middle

Time Complexity: O(1)

Space Complexity: O(1)

```
public static class LinkedList {
```

```
    Node head;
    Node tail;

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

    public void delete(Node n, Node prev) {
        if (n == null)
            return;

        if (n == head)
            head = n.getNext();
        else if (n == tail)
            tail = prev;

        if (prev != null)
            prev.setNext(n.getNext());
    }
}
```

Level: Easy

Follow Up: Given a node N in a Linked List, can you delete it without the previous node in O(1) time?

Questions to Clarify:
You should ask about the caveats mentioned below.

Solution:
We saw earlier that without the previous node, we can delete N in O(n) time. This was by iterating the list and finding the previous node.

To do it in O(1) time, there is a trick. We can copy over the next node's value into N, and then delete the next node.

Two caveats with this approach:
1. You are not really 'deleting' the node N. You are changing its value. You need to make this clear to the interviewer.
2. You cannot delete the tail node this way(there is no next node to copy from).

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.)

delete(head, tail, N)
     if N is tail, return
     copy value from N.next
     delete N.next
```

Test Cases:
Edge Cases: List Empty, N is tail, N is head
Base Cases: Single Item List
Regular Cases: N is in middle

Time Complexity: O(1)

Space Complexity: O(1)

```
public static class LinkedList {
    Node head;
    Node tail;

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

```
    }

    public void deleteWithoutPrev(Node n) {
        Node next = n.getNext();
        if (next == null)
            return; // cannot delete

        n.setData(next.getData());
        delete(next, n);
    }

    /* Same function we implemented in first problem */
    public void delete(Node n, Node prev) {
        if (n == null)
            return;
        if (n == head)
            head = n.getNext();
        if (n == tail)
            tail = prev;
        if (prev != null)
            prev.setNext(n.getNext());
    }
}
```