Remove Elements from a Linked List

The next important method that any implementation of a linked list will need is a `remove` method. This method should take the element we want to remove as an argument, and then search the list to find and remove the node that contains that element.

Whenever we remove a node from a linked list, it's important that we don't accidentally orphan the rest of the list in doing so. Recall that every node's `next` property points to the node that follows it in the list. If we're removing the middle element, say, we'll want to make sure that we have a connection from that element's previous node's `next` property to the middle element's `next` property (which is the next node in the list!)

This might sound really confusing, so let's return to the conga line example so we have a good conceptual model. Picture yourself in a conga line, and the person directly in front of you leaves the line. The person who just left the line no longer has her hands on anyone in line--and you no longer have your hands on the person that left. You step forward and put your hands on next person you see.

If the element we wish to remove is the `head` element, we reassign the `head` to the second node of the linked list.

Write a `remove` method that takes an element and removes it from the linked list.

Note: The `length` of the list should decrease by one every time an element is removed from the linked list.

⚗	Your `LinkedList` class should have a `remove` method.
⚗	Your `remove` method should reassign `head` to the second node when the first node is removed.

🧪	Your `remove` method should decrease the `length` of the linked list by one for every node removed.
🧪	Your `remove` method should reassign the reference of the previous node of the removed node to the removed node’s `next` reference.
🧪	Your `remove` method should not change the linked list if the element does not exist in the linked list.