








Create a Doubly Linked List

All of the linked lists we've created so far are singly linked lists. Here, we'll create a *doubly linked list*. As the name implies, nodes in a doubly linked list have references to the next and previous node in the list.

This allows us to traverse the list in both directions but it also requires more memory to be used because every node must contain an additional reference to the previous node in the list.

We've provided a `Node` object and started our `DoublyLinkedList`. Let's add two methods to our doubly linked list called `add` and `remove`. The `add` method should add the given element to the list while the `remove` method should remove all occurrences of a given element in the list.

Be careful to handle any possible edge cases when writing these methods, such as deletions for the first or last element. Also, removing any item on an empty list should return `null`.

	The <code>DoublyLinkedList</code> data structure should exist.
	The <code>DoublyLinkedList</code> should have a method called <code>add</code> .
	The <code>DoublyLinkedList</code> should have a method called <code>remove</code> .
	Removing an item from an empty list should return <code>null</code> .
	The <code>add</code> method should add items to the list.
	Each node should keep track of the previous node.
	The first element should be removable from the list.



The last element should be removable from the list.