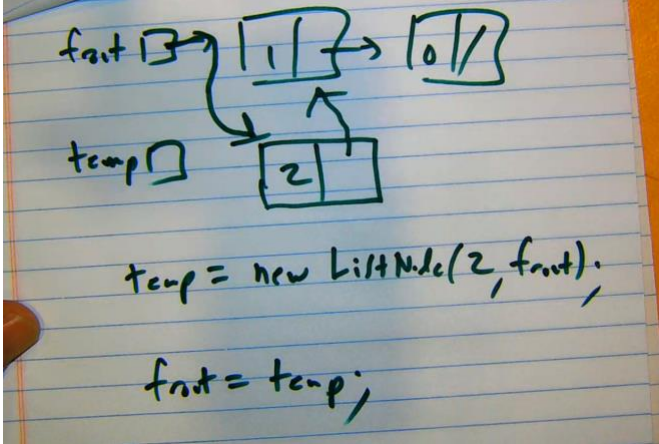


CSE143 Lecture Questions for Friday, 4/16/21

Question	Answer
<p>At this point, temp is supposed to have an arrow pointing to the node with 2 in it right? Ah ok! Yeah :-).</p>  <p><code>temp = new ListNode(2, front);</code> <code>front = temp;</code></p>	<p>Yes, there should be an arrow pointing from temp to the node with 2 in it. Too bad we didn't have a live lecture where you could have asked.</p>
<p>I'm still a bit confused with this, for example on practice it problems, how do we know if we set <math>q = p</math> or <math>p = q</math>? Is there an easy way to tell or is it dependable?</p> <p>Got it, I'll definitely go rewatch that lecture, thanks!</p>	<p>I mentioned in Monday's lecture that you can number the different places where a ListNode reference can be stored and by comparing the before and after pictures, you can figure out which need to change. If p is one of the things that changes, you may need to set it to q, and vice versa.</p>
<p>To find the spot to insert the sorted number, why would <math>\leq</math> be more efficient? Wouldn't that imply going right to the 'front' of the list of repeated values and not stopping as you encounter the first one? Okay. Thanks!</p>	<p>You're right...I think I said the wrong thing. We'd want strictly less than so that it stops when it finds the first one. The notes have it right.</p>
<p>You got cut off a bit when talking about a prev, will you explain more in a future lecture or video?</p>	<p>The lecture notes have a bit more explanation, but there isn't a lot more to say. It can simplify your code to keep track of a prev and you are allowed to use that approach if you prefer it.</p>
<p>So some of the cases we covered today are middle, end, front, and empty. What are some other cases that we should be looking out for? Ok thanks.</p>	<p>Those are classic cases that can apply in any situation. Other cases would be relevant depending on the specific problem you are solving.</p>

<p>I'm confused on the concept of the short circuit evaluation. Why would our code fail if the tests in an if statement are in the wrong order?</p>	<p>In the code we wrote, I originally wrote it this way:</p> <pre>if (value &lt;= front.data    front != null) {     ... }</pre> <p>That won't work. Think of what happens when front is null. We're testing for it, but before we get to that test, we see whether value is less-than-or-equal to front.data. That causes a NullPointerException because front is null. So it's important to put the other test first so that we only perform the comparison against value when we know that front is not null.</p>
<p>28:26 Shouldn't you have excluded the equal sign in the while loop to make it more efficient since it would keep going to the end of the line if you included it?</p> <p>----</p> <p>Sorry to comment on another student's question. I'm a bit confused b/c this is contradicting your reasoning to include an "=" sign. So would you include the equal sign or not?</p> <p>thanks!</p>	<p>Yes. Good observation. The notes and handout #4 have it right.</p> <p>It's easy to get confused because we tend to reason about why a loop should terminate. We'd want to stop moving forward if we get to a value that is equal to the one we are trying to insert. But the while loop has a test for why to continue, not why to stop. So the right way to do that is to use strictly less-than rather than less-than-or-equal in the while loop test. That way it stops if it finds a value equal to the one being inserted rather than skipping forward to get to the end of the sequence of duplicate values.</p>
<p>If you have more than one constructor, and they're very different (ex: one has a for-loop, and the other one just initializes), does one constructor still have to call the other one?</p> <p>Thanks!</p>	<p>We don't have to have one constructor calling another. We do it just to eliminate redundancy. That would only be helpful if the two constructors are very similar.</p>
<p>Can you make the back of a LinkedList point to the front of the LinkedList?</p>	<p>Yes, that would form what we call a circular list. It's a different approach that is sometimes used.</p>
<p>Are LinkedLists related to graph theory? That's what they remind me of</p>	<p>Graphs are a more general kind of structure than a linked list. You could think of a linked list as a very specialized and constrained graph.</p>