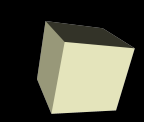
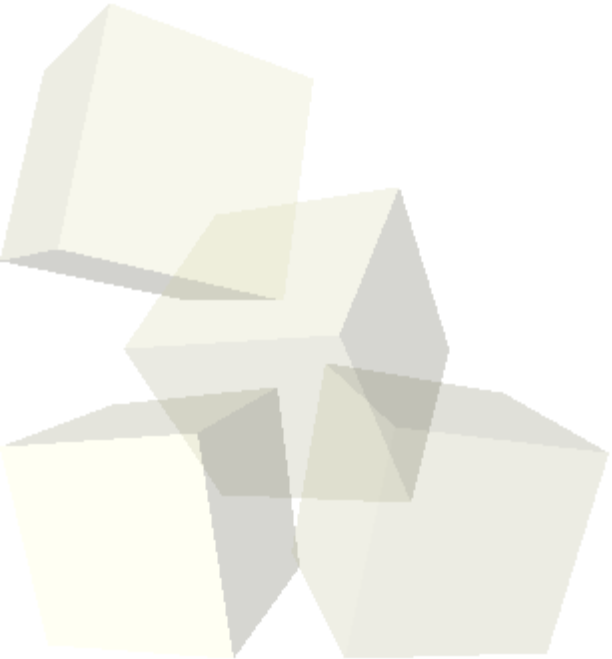


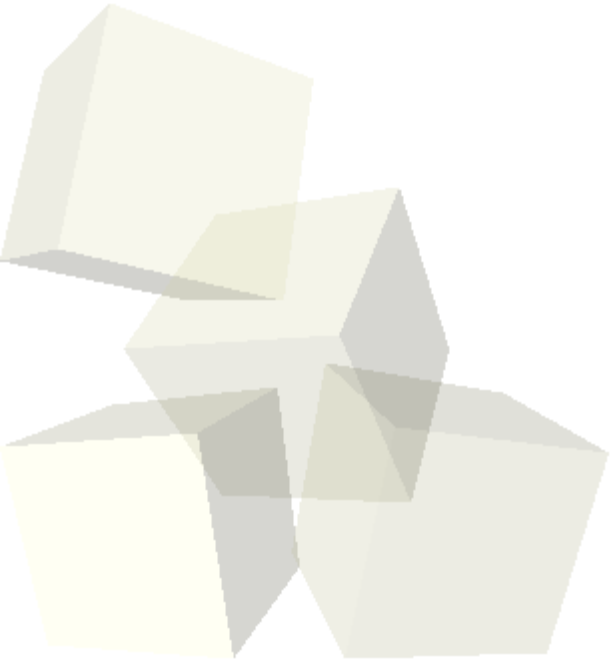
Linked Lists

2-16-2006



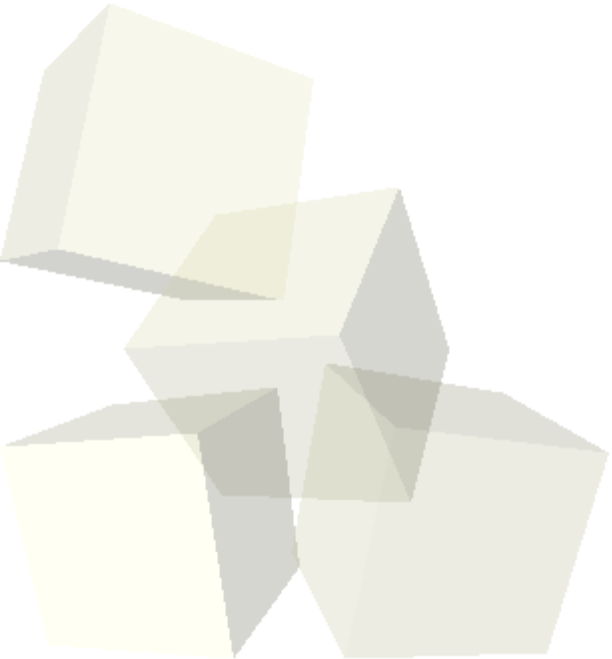
Opening Discussion

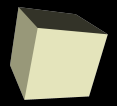
- What did we talk about last class?
- Do you have any questions about the reading?
- Do you have any questions about the assignment?
- Making a parallel sort.



Growing Stacks and Queues

- Before we go into the main topic for today, let's review our code for the array based stack and queue. What we want to add today is the ability for the arrays to grow if we put in too many elements.



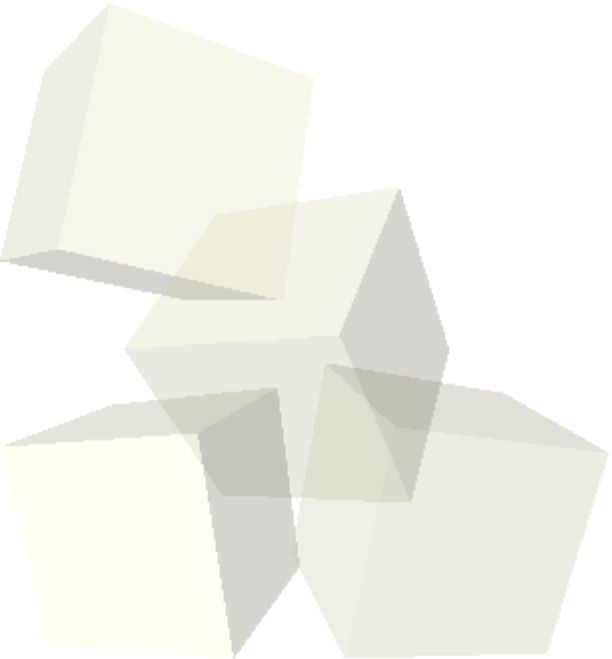


The List ADT

- Last time we talked about stack and queue as ADTs. They tell you what happens when different operations are invoked, but they don't tell you how it happens.
- The stack and queue are extremely simple. Today we move up to the List ADT. This is much like you writing a list on paper. You have the ability to add and remove random elements based on their location in the list or by specifying the item itself.
- Let's look at the java.util.List interface for an example. We won't try to write any lists that complex.
- Obviously you could implement the list ADT with an array.

Linked Lists

- Another way to implement the list ADT is with a linked list.
- What is a linked list? Why would you use one? What are the strengths of a linked list? What are the weaknesses of an array based list?
- How do we add to a linked list?
- How do we remove from a linked list?

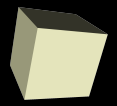


Variations on a Theme

- Your book only mentioned singly linked lists and gives the impression all linked lists are singly linked. This is not true.
- Linked lists can be doubly linked so each node points to the one before it and behind it. The advantage here is that you can remove or insert a node without keeping track of the one before it.
- There are also many uses for circular linked lists. These can be singly linked or doubly linked. In a circular linked list, the tail points back around to the head (and the other way around if it is doubly linked).

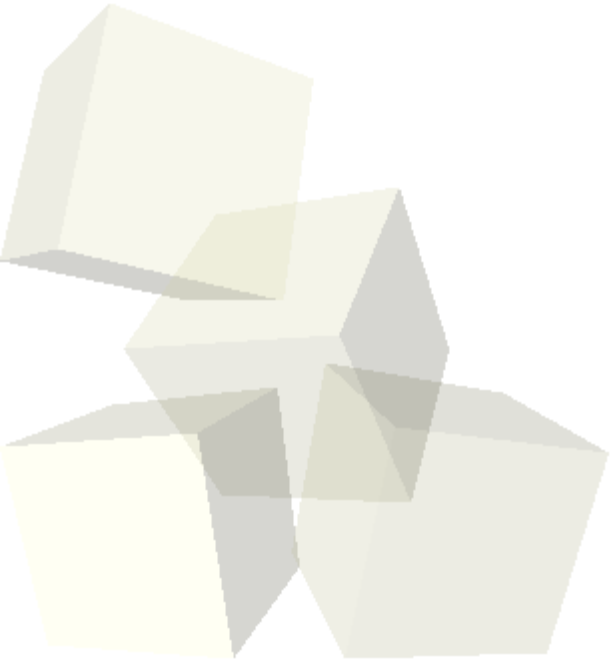
Sentinels

- Another feature your book doesn't mention is the use of a sentinel. This is a node that is in the list (typically a circular list), that doesn't hold data. It simplifies the code by removing special checks for null pointers.
- Writing a doubly linked list without a sentinel is a decent challenge because there are lots of if statements that have to make sure a link isn't null. With a sentinel the code is greatly simplified.



Code

- Let's write a doubly linked list that uses sentinels. We'll also have to decide on a List interface that we want to use. We could try using the Java one and realize we probably won't implement all the methods yet.



Minute Essay

- Why would you pick a linked list over an array based list?
- The design for assignment #3 is due on Tuesday.