Ch 20 Linked List

20.1 Intro

Definitions:

- successor - the element after the current element in a list
- predecessor - the element before the current element in a list
- contiguous allocation - large memory chunk, as used in an array
- linked allocation - non-consecutive memory, allocated with each element, as in linked list
- node - object that stores a list element and a reference to the next node in the list
- head - the first element in the list
- tail - the last element in the list

Basics in `LinkedList0` example on page 1174: create lists, add nodes to end of list, insert nodes in middle of list, removing a node, traversing a list…

/* the concepts are easy, the code is detailed and error-prone to write the first time */

20.2 Linked List Operations

Most basic methods:

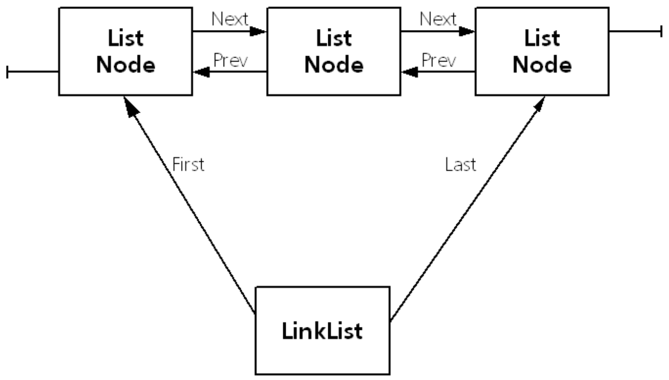
```
	public interface LinkedList1Int {
		boolean isEmpty();
		int size();
		void add( String e);   // add to end
		void add( int index, String e);   // add at position
		String remove( int index);   // remove by position
		boolean remove( String e);   // remove by value
	}
```

20.3 Doubly-linked and circularly-linked Lists

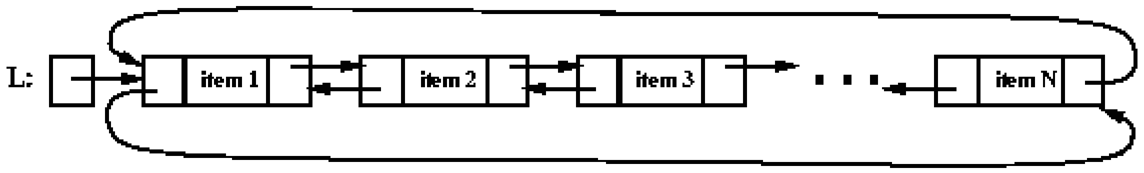
In a doubly-linked list, each node has a next and previous node pointer.
In a circular list, the last node's pointer is the first node and (if doubly-linked), the first node's previous pointer is the last node.

See doubly-linked list example in book: page 1191.

Doubly-linked list:



Circular linked list:



20.4 Recursion and Linked Lists

“A linked list is an inherently recursive data structure” -page 1197