Test 1 will be Thursday, Feb. 28, in class. It will be closed-book and notes, except for one 8.5" x 11" sheet of paper containing any notes that you want. (Yes, you can use both the front and back of this piece of paper.) The test will cover the following topics (and maybe more).

Chapter 1. Introduction
Motivation - Why study data structures? Why study algorithms?
Abstract Data Type concept - information hiding, encapsulation, implementation-independent view of data
Python Basics:
Primitive classes: int, float, long integer, boolean
Built-in collections: strings, tuples, lists, dictionaries
Assignment as a changing a reference
Control structures: if, while, for, break, continue, return
File operations: open, close, read, readline, readlines, seek, write, flush
Definition of functions, modules, and classes; scope rules in Python
General idea of event-driven programming using Python Tkinter GUI module, mouse and keyboard events

Chapter 2. Linear Data Structures (and Section 7.2 on link lists)
General concept of a stack: LILO, top and bottom
Stack Operations: pop, push., peek, size, isEmpty, isFull
Stack Implementations: List-based (Listings 2.1 and 2.2 in text) and linked using Node class (Listing 7.1)
Big-oh notation for each of the stack operations on any implementation
General usage of stacks: run-time stack as a program executes, balancing parentheses and Infix-to-Postfix conversion, postfix evaluation

General concept of a queue: FIFO, front and rear
Queue Operations: enqueue, dequeue, front, size, isEmpty, isFull
Queue Implementations: List-based (Listing 2.9 in text) and linked using Node class (Listing 7.1)
Big-oh notation for each of the queue operations on any implementation
General usage of queues: simulation of waiting lines

General concept of a Deque: add and remove from either the front or rear
Deque Operations: addFront, addRear, removeFront, removeRear, size, isEmpty, isFull

General concept of an unordered list: linear but we can access any item
Operations on unordered list: add, remove, search, length, isEmpty, isFull
General concept of an ordered list: linear of arrangement based on sorted order of items, an ordered list IS_A subclass of an unordered list, so we can use inheritance
List Implementations: built-in Python List via arrays, and linked list using Node class (Listing 7.1)
Big-oh notation for each of the list operations on any implementation

Chapter 3. Recursion (and Section 7.3 on Dynamic Programming)
General concept of recursion: split a problem into smaller and smaller subproblems until the solution is trival (the base case(s)).
Implementation of recursion via the run-time stack of call-frame which store the return address, parameters, and local variables

"The Three Laws of Recursion" (revised from the text to high-light the "divide-and-conquer" problem solving technique) : (1) a recursive algorithm must have one or more base cases, (2) a recursive algorithm must divide the original problem into smaller problem instance(s) of itself which is solves recursively, and (3) knowing the answer to the smaller problem instance(s) you must be able to combine their answers to "conquer" the original problem.
General usage of recursion on simple problems like factorial, fibonacci, binary search
Understand the problem with some divide-and-conquer algorithms, such as fibonacci, that solve the same smaller problems many times
General concept of the dynamic programming problem-solving technique:

- View the problem recursively as in divide-and-conquer, but solve instances of the problem from the base case(s) toward the larger problem of interest (i.e., the original problem you're trying to solve)
- Store the answers to the smaller problems and lookup their answer if needed when solving a larger problem instance (this way you solve a problem instance only once)

General usage of dynamic programming on simple problems like fibonacci and binomial coefficient (lab 5)

Chapter 4 (only sections 4.1 to 4.3)
Understand why algorithm analysis is important with respect to execution time and memory usage
General definition of big-oh notation
Types of big-oh analysis for best-case, worst-case, and average-case
Be able to use big-oh notation to describe execution time (e.g., big-oh of the pop stack operation given a specific stack implementation)
Common functions for big-oh: O(1) constant time, O(log n) logarithmic, O(n) linear, O(n log n) log linear (I always say "n log n"), $O(n^2)$ quadratic, $O(n^3)$ cubic, $O(2^n)$ exponential, and O(n!) factorial.
Searching algorithms: sequential and binary search (understand both the iterative and recursive algorithms)
Understand that an
General concept and goal of hashing
Hashing terminology: hash table, slots, hash function, remainder method, collision, load factor
Open address vs. chaining (closed address) hashing
Rehashing techniques: linear probing, quadratic probing
General concept of primary and secondary clustering