Exam 1 Study Guide

Dynamic Memory and pointers:

- □ pointers (syntax, meaning)
- □ dynamic allocation using new
- □ freeing memory using delete
- □ allocation of 1- and 2-dimensional arrays
- □ memory leaks and dangling pointers
- □ pointers as arguments (by value, by reference)
- □ const and pointers
- □ member-selection operator (->) as alternative to (*ptr).member

Functions:

- □ general syntax
- □ pass-by-value and pass-by-reference arguments
- □ prototypes

Classes:

- □ basic syntax (be comfortable reading class declarations)
- □ visibility modifiers (public, private, protected)
- □ static modifier and what it means
- □ constructors
- □ initialization lists – setting members and calling base-class constructors
- □ inline methods (syntax and how the compiler treats them specially)
- □ const member functions
- □ operator overloading (global, friended, and member versions)
- □ requirements for classes using dynamic memory
- □ the big 3 (copy constructor, assignment operator, and destructor)
- □ behavior of automatic copy constructor, assignment operator, and destructor
- □ assignment operator best practice (chaining, self-assignment)
- □ structs (essentially a class, but defaults to public visibility)
- □ inheritance (syntax, meaning, what it does and how it does it)
- □ automatic behavior of constructors (calls base class default constructor, then member constructors, etc); requires base class to have default constructor!
- □ normal method of overriding methods base-class functions
- □ virtual methods and pure-virtual methods of overriding base-class functions

Namespaces:

- □ purpose and benefits
- □ how to access items in a namespace (e.g., using namespace std; or std::string)

Container classes

- □ general idea of the methods the bag class has
- □ general idea of how each method is implemented
- □ complexity analysis of methods using Big-O syntax
- □ understand the differences between the static-array, dynamic-array, and linked-list versions

Linked Lists:

- □ general idea and structure (singly-linked lists only)
- □ the NULL pointer and how it is used in lists
- □ understand the common components of a node (data and pointer to next node)
- □ traversing a linked list
- □ inserting at the head of a list
- □ advantages and disadvantages of a linked list over an array

Miscellaneous:

- □ scope-resolution operator (what it does; how it's used)
- □ typedef
- □ the this keyword
- □ pointers and dynamic memory—seriously! =)

General tips:

- you will be asked to write a fair amount of code by hand; practice using a pencil and paper
- be comfortable reading code, too; practice reading through the code in the book
- speaking of the book, make sure you're read the assigned sections...
- know how to write a for loop, or you're screwed... (you'll probably need to write four of them)