Here’s a (partial) list of topics and review problems for the second test. Use this as a reference to begin preparation for the second test.

Chapter 5: Recursion & Trees

1. Vocabulary & Definitions
   (a) Recursive Algorithm
   (b) Recursive Method
   (c) Postfix, Infix Expressions
   (d) Dynamic Programming: Top Down (Memoization) & Bottom Up
   (e) Graphs, Nodes, Vertices, Edges, Adjacency List, Adjacency Matrix
   (f) Trees, Binary Trees, Internal Nodes, External Nodes, Leaves, Root Node
   (g) Tree Traversal: Preorder, Inorder, Postorder, Levelorder (Top Down & Bottom Up)
   (h) Graph Traversal: Breadth First Search, Depth First Search

2. Concepts, Skills, & Theory
   (a) Be able to recognize, analyze, and write a recursive algorithm.
   (b) Understand how to use dynamic programming (top down or bottom up) techniques to improve the performance of a recursive algorithm.
   (c) Understand the differences between Top Down and Bottom Up dynamic programming.
   (d) Be able to create the recursive call tree associated with a recursive method.
   (e) Understand the basic concepts, definitions, and properties of graphs and trees.
   (f) Be able to translate an Adjacency List (or Matrix) into a Graph and vice versa.
   (g) Given a Tree, be able to determine the order nodes are visited in the preorder, inorder, postorder, or levelorder traversal methods.
   (h) Given a Graph, be able to determine the order nodes are visited in breadth first search or depth first search.

Chapter 6,7: Searching

1. Vocabulary & Definitions
   (a) Stable and Unstable Sorting
   (b) Adaptive and Nonadaptive Sorting
   (c) Insertion, Selection, Bubble Sort
   (d) Quicksort & its variants
   (e) Mergesort & its variants

2. Concepts, Skills, & Theory
   (a) For each of the sorting methods covered in class (listed above), understand the underlying algorithm and its performance characteristics (approx number of comparisons and exchanges).
   (b) Understand the best, average, and worst cases for each of the sorting methods.
(c) Understand particular situations where each sorting method are either very good or very poor. Combinations of functions.

**C Programming & Memory Layout**


1. Vocabulary & Definitions
   (a) Pointers & Addresses
   (b) Stack and Heap Memory

2. Concepts, Skills, & Theory
   (a) Be able to describe some of the major differences between the Java & C programming languages, as well as some of the strengths and weaknesses of each (e.g., OO vs. Procedural, garbage collection, mechanisms for parameter passing, well-defined standards, platform independence, etc.)
   (b) Understand the purpose of pointers and be able to recognize and describe situations where they can, should, and should not be used.
   (c) Understand how to access and use the address of a variable.
   (d) Understand the difference between Local (Stack) Memory and Heap memory as well as the advantages and disadvantages of each.
   (e) Be able to interpret and write simple C programs, including C programs that involve pointers and that utilize both Local and Heap memory.

**Homeworks**

1. Concepts, Skills, & Theory
   (a) Be able to describe the steps involved in the translation of an expression from infix to postfix.
   (b) Be able to describe the steps in evaluating a postfix expression on a stack.
   (c) Be able to compare and discuss different approaches to solving the ParkingLot problem (2 from class, 1 from homework).