Midterm study sheet for CS2711

Here is the list of topics you need to know for the midterm. For each data structure listed below, make sure you can do the following:

1. Give an example of this data structure (e.g., draw a binary search tree on 5 nodes).
2. Know the basic properties of data structures (e.g., that a heap has height $\log n$) and be able to prove them by induction.
3. Know which basic operations the data structure supports and be able to show them on an example (e.g., insertion into an AVL tree).
4. Know what is the time complexity of its basic operations and how they compare between different structures (e.g., searching in an AVL tree vs. searching in a linked list).
5. For each algorithm, make sure you know its time complexity, can write pseudocode for it and can show its execution on an example.

For the analysis of algorithms (chapter 4), you need to know time complexity and induction, and be able to solve problems similar to ones in labs and assignments. Also, have a basic understanding of amortized analysis (for splay trees and doubling array size) and probabilistic analysis (hash tables, skip lists, quicksort).

List of topics

1. **Basic data structures (chapters 3 and 6)**: Linked list, doubly linked list, array, extendable array. Know why doubling the size is a better way to grow an array.
2. **Stacks and queues (chapter 5)**: Know stack and queues data types (push/pop, enqueue/dequeue). Be able to do parentheses/tag matching using stacks.
3. **Trees (chapter 7)**: Know both general trees and binary trees. Terminology like root/parent/child, height/depth/size,leaf/internal node, proper binary tree, complete tree. Traversals (preorder, inorder, postorder). Evaluating arithmetic expressions using postorder traversal, drawing trees using inorder traversal. Euler tour of a tree. Linked-list and array representations of trees.

   Know the relationships among tree height, the number of nodes, number of edges, number of leaves, etc and be able to prove them (e.g, by induction).
4. **Heaps (chapter 8)**: Know the definion of a heap and a priority queue, both bottom-up and top-down heap construction. Using arrays to store heaps. Heapsort.
5. **Hash tables (chapter 9)**: Know the definition of a hash table. Collision-handling schemes: chaining, linear probing, quadratic probing, double hashing. Be able to give an example of a hash function and a hash table, and state which properties a good hash function should have (output "looks random", not likely to get a collision). Know polynomial hash codes as well as $h(x) = ax + b \mod n$ function class.
6. **Skip lists (chapter 9)**: Know the definition and performance of a skip list. Have an intuition why skip list is efficient on average, why it has on average $\log n$ layers. Know which additional operations (successor, minimum) a skip list implements efficiently.

7. **Search trees (chapter 10)**: Know the definitions and properties of a generic binary search tree, as well as AVL tree, splay tree, (2,4) tree and red-black tree. Know the relationship between (2,4) trees and red-black trees (but I will not ask much more about (2,4) and red-black trees, so concentrate on AVL and splay trees). Be able to do AVL tree rotations (insert/delete) and the "splaying" operation.

8. **Sorting (chapter 11)**: For all sorting algorithms, know how they work (i.e., be able to show on a given input how they work), and their running time (whether average or worst-case and examples for worst/best cases). Know when you would use each of them (i.e., conditions when you would choose radix-sort, the fact that quicksort is in-place, etc). Know the lower bound fo sorting. Know how to adapt quicksort to do selection.