CSC 403 Final exam Study guide.

Disclaimer: This sheet is intended to be a representative review for the exam. However, you are responsible for all material that we have covered and all the assigned readings, whether or not they are explicitly referenced here.

The final exam is comprehensive, testing all material covered during the quarter. Approximately 60% of the final will focus on material covered after the midterm exam.

The exam will be closed book, notes, no calculator.

Format: The exam may consist of any of the following:

- Coding,
- Multiple choice,
- Fill in the blank,
- short answer,
- analysis,
- draw the data-structure given the input.

Study suggestion:

- See the objectives for weeks 6-10.
- Review the midterm study guide & midterm exam
- Review the sample questions on this study guide.
- Review the daily ICE exercises.
- Review the quizzes in D2L
- Review your HW solutions.
- Attempt the suggested exercises in the daily todo lists

Sample and study questions:

1. Reflect on the coding assignments: What were the 'lessons learned'?

2. For each major algorithm/data structure, identify a 'real' problem/application which it 'solves'.

3. Why is there a whole chapter of the book focused on string algorithms?

4. Consider the (adjacency-list) basic Graph data structure from the text shown (supplied at test time:*satt*).   We wish to add a function to:
    a. Delete a node from the graph.
    b. Delete an edge from the graph
    c. Add a node to the graph
    d. Compute the maximum degree
    e. Compute/return the complement of the graph

    a) Give an outline of the function
    b) Analyze the complexity of this operation in terms of V, E
    c) Note any relevant observations

d) How would your answer change if the underlying representation was an adjacency matrix?

   a. Example: Connected Components
   b. Example: Eccentricity
   c. Example: topological order
   d. Example: Red-Black tree

5. Outline an algorithm to determine if a graph has a cycle. Estimate the complexity.

6. Compare and contrast bfs, dfs. Discuss the role of these frameworks in creating graph algorithms. Describe how bfs/dfs was used in graph algorithm that we discussed.

7. The KS algorithm uses two passes of bfs. Explain what each pass contributes to the solution of the problem.

8. Using the author's graph design pattern, create a Java class to perform the following:
   e. Determine if a graph is connected
   f. Determine if the graph has a cycle
   g. Perform BFS – labeling vertices in the order visited
   h. Perform DFS – labeling vertices using inorder (postorder)

9. Discuss the Erdos-Renyi random graph model and its role in the HW7B experiment.

10. Explain how key-indexed counting can be used to sort an array of strings. What is the complexity? How is this algorithm affected by the size of the alphabet?

11. Apply the LSD string sorting algorithm to the array of strings shown (satt). Show the resulting array after completely processing the first two character positions (0 and 1).

12. Compare and contrast 'normal' quicksort to 3-way string quicksort.
13. Compare/contrast the string sorting algorithms from 5.1
14. Compare/contrast the String-key specific ST from 5.2 with ST implementations from chapter 3.
15. Consider the table on page 784. Be able to write a short justification for the order-complexities for each data structure considered.
16. Draw the R-way trie that results when the given keys (satt) are inserted into an empty trie.
17. List all the key-value pairs stored in the given trie (satt).
18. Draw the TST that results when the given keys (satt) are inserted into an empty trie.
19. List all the key-value pairs stored in the given TST (satt).
20. Using the Java class definition for a trie, write a non-recursive version of the get (put) function.
21. Using the Java class definition for a TST, write a non-recursive version of the get (put) function.