1. Give the formal definitions for Big-O, Omega, and Theta from the Cormen text.

2. What is the Big-O for this code (assume that n is positive).
   ```c
   int f(int n)
   {
     if (n > 0)
       return 1 + n*f(n - 1);
     else
       return 5;
   }
   ```

3. Sketch pseudocode for Dijkstra’s algorithm

4. Sketch a data structure to represent a very large sparse graph.

5. Given a graph... Run Dijkstra’s algorithm by hand, to show the vertices visited, the final distance to each vertex, and the order that you visit them. Start from vertex A.

6. Sketch the connection between a vertex in a graph for Dijkstra’s algorithm, and an element of a binary heap, so that you can quickly reduce the distance of the vertex (and update the heap). This “extended heap” technique is a key part of making Dijkstra’s algorithm run quickly.

7. For Kruskal’s Algorithm, which edges would be in the spanning tree for the graph above? And in particular, WHICH ORDER would they be added to the tree?

9. Suppose you have the following array (indexed from 0) that is used to represent the “parent” of an element in a disjoint set: \{0, 2, 0, 3, 0\}. There are five vertices. How many sets are there? Show the “tree” representation that illustrates the two sets.

10. Draw a Venn diagram that shows the (most likely) relationship between P, NP, and NP-Complete. Also, give short definitions for P, NP, and NP-Complete.

11. What is meant by a “decidable” problem? Include the term “certificate” in your description.

12. (10 points) Convert the 3CNF satisfiability problem \((A+B+C)(A+B+C)(A+B+C)\) into a subset-sum problem.

13. The subset-sum problem is really the check cashing problem that we solved earlier in the semester. What makes our check-cashing “pseudo-polynomial?” Why can’t we simply use our check-cashing code to solve all subset-sum problems (and thus, all 3CNF problems)?

14. Convert the 3CNF problem in the previous question into a maximum clique problem.

15. Sketch pseudocode to insert a value into a heap. You can assume that there are macros for LEFT, RIGHT, and PARENT.

16. Construct an optimal binary search tree (and show your work) for the following values:
   A - 5   B - 11   C - 2   D - 8   E - 3

17. For Quicksort, we looked at two different partitioning functions, one by Lomuto, and a second one by Hoare. Explain how randomization is useful for both (and what problem it prevents) -- you might want to give an example.

18. (3 points) Sketch pseudocode for the Quicksort routine, assuming that you’re using the Hoare partition function. Note that this is different from what you might do with the Lomuto function (and there’s a small, but very important difference).