S-Q 1997 - Practice Hourly 1

This practice hourly is intended as a study aid for the first S-Q hourly. There are many more problems here than will actually on the midterm, so this is not meant to represent the same length of test as the midterm. Each problem is assigned a point value that roughly corresponds to the number of minutes that I expect you would spend on that problem during the midterm. (The same point scheme will be used on the actual midterm.) Let me know if I am wildly wrong on these. (I suspect many of the harder problems deserve more time.)

- Many of the problems are given with hints. On the hourly, there will generally be fewer hints.
- The problems are divided into several categories; you should expect to find at least one problem from each category on the hourly.
- The problems from the Weiss book are good review, but many are harder than what will be on the midterm.

1 Background

1. (5 points for part a, 10 points for part b) Weiss, 1.10. Use induction. Part a should be straightforward. Part b uses exactly the same method, but a little more algebra is necessary.
2. (10 points) Weiss, 1.8. Use the exp algorithm from asst1, and use **mod** early and often. Should only require a handful of multiplications, all of very small numbers.
3. (20 points) Weiss, 2.1. This is tedious, but shouldn't present any major difficulties.
4. (12 points) Weiss, 2.2. Give counterexamples.
5. (12 points) Weiss, 2.6. (Don't actually implement them, of course.) The first four shouldn't take more than a minute each, but the last two are harder.
6. (5 points) Weiss, 2.20. Rephrase the question as:

   In the context of finding the big-O for a piece of code, why is it important that integers in our computer model have a fixed size?

2 Lists, Stacks, and Queues

1. (5 points) What is an abstract data type? What is a data structure? What is the difference? Give at least one example of each.
2. (15 points) Weiss, 3.5. The notation $L_1 \cup L_2$ means "all the elements in L_1 or L_2" (but only one instance of each– the result must have no duplicated values.)

There is an algorithm that runs in $O(n)$ time (where n is the length of the longest list) for linked lists (actually, a tighter bound can be placed on the running time if the lists are of unequal length). This is not unlike something we did in class.

3. (10 points) Weiss, 3.11. Just write the code iteratively and recursively; forget the part about crashing the program.

4. (10 points) Weiss, 3.17. Be specific about the advantages and disadvantages.

5. (10 points) Weiss, 3.21. Push and pop must both take time $O(1)$. Not as silly as it might look; many systems actually use this trick in some way.

6. (10 points) Weiss, 3.23. I don't think that push and pop can be done in time $O(1)$ (but note that this wasn't required by the question), but feel free to surprise me.

3 Binary Trees

The first hourly will cover some general tree knowledge, including binary trees and binary search trees. It will *not* cover AVL, 2-3, B-trees or tries. As a general theme, brush up on your recursive traversals.

1. (5 points) Write an algorithm that searches for a key in an binary search tree.

2. (5 points) Write an algorithm that searches for a key in an **unordered** binary tree.

3. (15 points) Write an algorithm that determines whether **both** keys A and B (where $A < B$) are present in a binary search tree. In the worst case, this might take as much time as searching for A and then searching again for B, but in the best case you should be able to save a considerable amount of time.

   Note that this question does not ask for an analysis of the expected running time– just identify the best and worst cases.

4. (10 points) Weiss 4.28.

5. (10 points) Weiss 4.40. Similar to sparse arrays, from the course book.

6. (15 points) Weiss 4.42. For the sake of simplicity, assume that the tree is binary (once you have a solution, you should be able to generalize to n-ary trees). This problem is probably harder to understand than it is to solve, but has a nice solution. There is a linear solution.

   (This is a special case of a problem that is computationally much more difficult, which we may meet later in the semester.)

7. (15 points) Weiss 4.44. Remember that the elements have unique keys; this may be important. Do not assume that the keys inserted into the tree are necessarily unique, however– if a duplicate key is inserted, it should replace the existing key. Be sure to describe any changes necessary to Insert and Delete.