CS 261 — Spring 2025 — Final exam (not comprehensive)

Name:

UCI email:

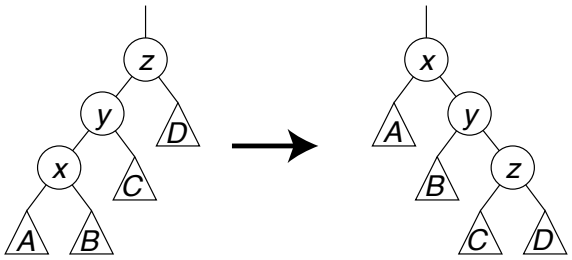
Do not open this exam until told to start by the instructor.

This is a closed book, closed note exam. Calculators or other electronic devices are not allowed.

Please write your answers ONLY on the front side of each page.
Answers written elsewhere will not be scanned and will not graded.

You may use the back sides of the pages as scratch paper.
Do not unstaple the pages.

1. (15 points) The following figure from Lecture 6a shows what is supposed to happen for a single step of splay(x), in the "zig-zig" case where x is a left child and left grandchild. If you start with the left tree in the figure, but then rotate x with its parent y, and then rotate x with its new parent z, something different happens. What does the tree look like after these two rotations?



2. (15 points) In the augmented binary search tree data structure for ranking and unranking queries, we augmented each node of the tree with the rank of its key among the subsequence of keys within its subtree (equal to the number of keys in the subtree of its left child). Explain why it would not work well to instead augment each node with the rank of its key in the whole sequence of keys.

3. (15 points) For the problem of searching for the position of a query value q in each list of a collection of sorted lists:

   (a) What is the reason to prefer using fractional cascading over doing a separate binary search in each list?

   (b) What is the reason to prefer using fractional cascading over precomputing a single list of all possible query answers and performing a single binary search in this list?

4. (15 points) Draw the Cartesian tree for the array of numbers [15, 8, 12, 16, 6, 3, 20, 10], and then list these numbers in the order given by an Euler tour of the tree.

5. (15 points) Suppose that tree T with n nodes has been stored in a data structure for level ancestors, and that as part of this structure we can look up the level of any node x in T by calling level(x) or find its ancestor at level i by calling ancestor(x,i). But what we actually want to do is to answer lowest common ancestor queries, and we cannot replace our data structure with one that finds lowest common ancestors directly. Describe a way to find the lowest common ancestor of two nodes x and y in T by performing only $O(\log n)$ level ancestor queries.

6. (15 points) One of the practice problems involved partially persistent union-find, using "union by size" but not using path compression. It described simplified fat nodes that only store one timestamp and one piece of persistent information for each node: the parent of the node, with a timestamp recording the version at which this parent was added. The other information in a node, including the tree size stored at each root node, was not maintained persistently. Explain why it is unnecessary for the root size information to be persistent. (Hint: which operations access this information?)

7. (15 points) For the word "giggling$":

   (a) Give the suffix array for this word (in its basic form, only including the starting index of each suffix). Use the alphabetic ordering for the letters appearing in this word, $ < g < i < l < n.

   (b) Draw a suffix tree for this word.

8. (15 points) State what it means for a dynamic graph algorithm to be fully dynamic.