#1. Peer Credit Assignment

1 Point Extra Credit for replying

Please list the names of the other members of your peer group for class 1/25 and 1/27 and the number of points you think they deserve for their participation in group work on the two days combined.

- You have a total of 6 points to allocate across all of your peers.
- You can distribute the points equally, give them all to one person, or do something in between.
- You need not allocate all the points available to you.
- You cannot allocate any points to yourself! Points allocated to yourself will not be recorded.

#2 Red-Black Tree and (2,4)-Tree Deletion (12 pts)

Preliminary Comments
In this problem we delve deeply into the CLRS code for tree deletion. The lecture notes were based on Goodrich & Tamassia’s textbook, because they show the correspondence of RBTs to 2-4 trees, which makes the former easier to understand as balanced trees. The CLRS version differs somewhat. You will need to read the CLRS text to answer this question.

The cases for insertion are similar between G&T and CLRS, but the terminology differs (e.g., what the letters w, x, y, and z refer to). The cases for deletion differ: G&T have 3 while CLRS have 4! Be careful because there are mirror images of every situation (e.g., is the double black node a left child or a right child?): G&T and CLRS may be describing the same situation with mirror image graphs.

The top level methods in CLRS for RB-INSERT (p. 315) and RB-DELETE (p. 324) essentially do binary search tree (BST) insertion and deletion, and then call "FIXUP" methods to fix the red-black properties. Thus they are very similar to the BST methods TREE-INSERT (p. 294) and TREE-DELETE (p. 298). The real work specific to RBTs is
in these fixup methods, so we will focus on them in these questions, but you should also study the top level methods to understand them as BST methods.

**Problems**

You may want to use the Google Drawing template provided in a separate document.

(a) **RBT as 2-4 Tree** (2 pts) Draw the 2-4 tree that corresponds to the RBT shown below.

(b) **Deletion** (4 pts): Delete key 2 from the red/black tree shown above, and show the deletion in the (2,4) representation.
- Show every state of the RBT tree, including after the BST-style deletion and after each case applied by RB-Delete-Fixup. Clearly identify the colors of the nodes.
- Also show the state of the 2-4 tree for each of these RBT states.
- If a double black node occurs (node x in CLRS), clearly identify which node it is.
- For each state change, identify both G&T case(s) from the web notes and the CLRS case(s) from the textbook that are applied in each of your steps.
- Your final diagram should show the RBT after RB-Delete-Fixup and the (2,4) tree representation that results.

(c) **More Deletion** (6 pts): Delete key 1 from the initial red/black tree shown above (NOT the tree that results from (b)), showing all steps as specified above.

---

#3 Red-Black Tree Height (6 pts)
(a) (4 pts) What is the largest possible number of internal nodes (those with keys) in a red-black tree with black height k? What is the height of the corresponding 2-4 tree? Prove your claims.

(b) (2 pts) What is the smallest possible number of internal nodes (those with keys) in a red-black tree with black height k? What is the height of the corresponding 2-4 tree? Prove your claims (you may use Lemma 13.1).

#4 Red Nodes in Red-Black Tree (2 pts)

Consider a red-black tree formed by inserting $n$ nodes with RB-Insert. Prove that if $n > 1$, the tree has at least one red node. *Hint:* Nodes are red when inserted (line 16 of RB-Insert). Show that if $n > 1$ one node must be red after RB-Insert-Fixup is complete.

(*It is not sufficient to say that the second node inserted will be red. You must show that some node remains red under all possible insertion sequences and the transformations that result.*)

#5 Sorting larger numbers (10 pts)

Suppose we want to sort $n$ integers in the range 0 to $n^4-1$ in $O(n)$ time.

a. (2 pts) Show that Counting-Sort is not an option by analyzing the runtime of Counting-Sort on this data. *Hint:* Identify the value of $k$ and invoke the existing analysis.

b. (4 pts) Show that unmodified Radix-Sort is not an option by analyzing the runtime of Counting-Sort on this data. *Hint:* Identify the value of $k$ for each call to Counting-Sort. Then identify the value of $d$, and invoke the existing analysis.

c. (6 pts) CLRS section 8.3 states that “we have some flexibility in how to break each key into digits” and gives a relevant Lemma. Using this as a hint, describe a modified Radix-Sort that would sort this data in $O(n)$ time and do the analysis to show that this is the correct runtime.