Guidelines: You may brainstorm with others, but please write up the answers by yourself. Acknowledge all collaborations and external resources used.

1. Show a correct representation (virtual tree) of the actual tree in Figure 5.5.

2. Show the result of findPathCost on the tree in Figure 5.4(b), including the changes to the data structure. (Assume a splay tree implementation.)

3. Show how to perform a single rotation in a join/split path in constant time (show the changes).

4. In a link/cut tree with $n$ vertices, what is the minimum number of heavy edges? What is the maximum? Describe when each of these occurs.

5. Describe how to implement the evert operation on a link/cut tree in $O(\log n)$ amortized time, assuming that you have a reverse operation on join/split paths that takes $O(\log n)$ amortized time. (reverse changes the direction of every edge in the path, so the old head is the new tail and the old tail is the new head.) For a description of evert, see the description on page 70 of RET, along with Figure 5.7.

6. (Grads only) In this question, you will determine how to efficiently implement the “reverse” operation on a join/split path implemented as a modified splay tree. You’ll need one extra bit per node in the splay tree, which will modify the splay tree’s semantics and operations. Flipping this bit at the root of the tree reverses the order. Flipping this bit at the root of a subtree reverses just the order of that subtree. Assume that bit=false corresponds to the usual symmetric order (left child precedes self, which precedes right child).

(a) Describe how to locate the tail of the list. (Hint: You’ll need to check the bit at each iteration of your traversal!)

(b) Describe how to recursively restore the tree to symmetric order in $O(n)$ time. (After executing your algorithm, the tree should represent the same order as before, but have every bit set to false.)

(c) Splay trees splay! Explain how to perform a single rotation in this modified splay tree. (Since double rotations can be composed of single ones, this suffices.) You may choose to change the bits of one or more nodes, as long as your operation still runs in constant time ($O(1)$).