Burton Rosenberg

Quiz

December 14, 1992. 2:00–2:50 PM

There are four problems for a total of 100 points. Show all your work, partial credit will be awarded. Write in the provided blue books, write and sign your name on each one. No notes, no collaboration. Good luck.

Name: _____

Problem	Credit
1	
2	
3	
4	
Total	

1. (25 Points.) A *single rotation* is a tree operation which has many uses. This and the following two problems will explore a sample application. Be sure to take the time to understand the following picture:

   Pictured is a *single rotation* at node y. The starting configuration is on the left. The node x has right child y, and the entire sub-tree to its left is labeled A. The entire sub-tree to the left of y is labeled B, the entire sub-tree to the right of y is labeled C. After the rotation, y has been brought up into the position where x was, x becomes the left child of y, and sub-tree B is disconnected from y and reconnected as the right child of x.

   A rotation maintains search tree order in the tree. If you don't see why, don't worry — we will not need this fact anyway.

   There is another form of single rotation when y is the left child of some node:

- Write a program that implements single rotation.

You can write carefully composed pseudo-code or full Pascal. If you write pseudo-code, make sure to address all the relevant data-structure issues.

HINT: To rotate at some node w, you will need to know the parent of w. You might also need to know the grandparent of w. You might want to pass all these things to your procedure.

2. (25 Points.) After single rotation, we naturally go on to *double rotation*. There are two forms of double rotation, *zig-zag* and *zig-zig*. Fortunately, double rotation can be implemented in terms of two single rotations, but you must tell me which two single rotations.

   (a) Here is a zig-zag at z:

How do you do this with two single rotations?

(b) Here is zig-zig at z:

How do you do this with two single rotations?

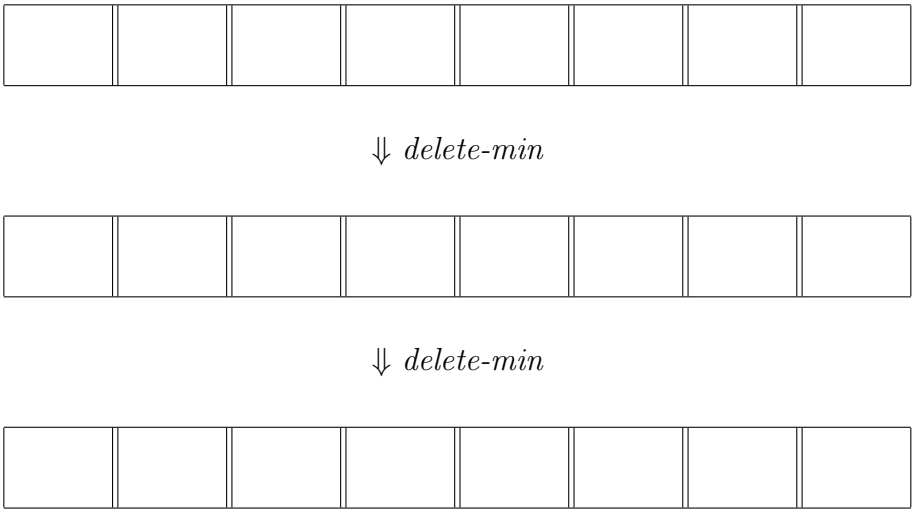
3. (25 Points.) Suppose we have a nice, binary search tree T. For simplicity, all the elements of the tree are distinct. Given a node x in the tree, we want to *split the tree at* x. This means, starting with T and x, we end up with two trees, T_1 and T_2, such that all nodes with labels less than the label of x are in T_1 and all nodes with labels greater than the label of x are in T_2.

   This is simple to do when x is the root. In this case we just delete the root leaving two trees, the left sub-tree of the root and the right sub-tree of the root. These would be T_1 and T_2, respectively.

   - Given a tree T and a node x in the tree, how do you split the tree T at x even if x is not the root.

   HINT: Use zig-zig, zig-zag and single rotation.

4. (25 Points.) What is the result of two consecutive delete-min's from the following heap:



The dash means that the array position is empty.