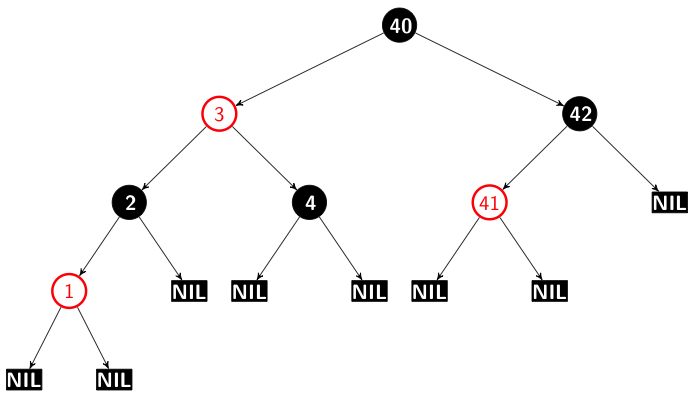
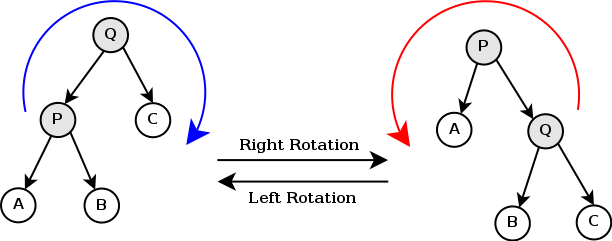
**Part 1: Implementing a Red Black Tree**

****

**Description:**

For part 1 you will need to implement a left-leaning red black tree (LLRBT) for storing the same String-Integer key-value pairs from Project 2 (i.e. <user id>, <score>). A red black tree is a self balancing binary search tree which, when following a specific set of rules, guarantees all search, balance, and delete operations are O(logn). Nodes in the tree now have an extra parameter for color which can be either red or black.

There are 6 rules for a LLRBT:

1. The root node is always black in color.
2. Every new node inserted into the tree is red.
3. Every null child of a node is considered black.
4. All red nodes should be left-leaning (i.e. left children.)
5. No path should have two red nodes in a row.
6. No node should have two red children.

To maintain these above rules, color swapping and rotations of nodes can be performed. For example, If rule 4 is broken, a left rotation should be performed and the colors of the current node and its left node should be swapped to follow rule 2 (for clarity, the color swapping is done after the rotation). If rule 5 is broken, a right rotation should be performed on the current node and the colors of the current node and its right child should be swapped to follow rule 2.

If rule 6 is broken, invert the colors of the current node, and both its children.

Insertion into the tree is the same as a regular binary search tree except after every insertion you must ensure that the tree follows the above rules.

**What’s included:**

You are given a completed binary search tree like the one in project two with the appropriate methods. Additionally, **methods for performing a right rotation, left rotation, and swapping the colors of two nodes have been given to you.** If you prefer to use your own implementation of a BST from Project 2, you may, but make sure everything works with the provided tests.

**What’s expected:**

You will need to modify the insert method so that all rules are followed to ensure it is still a valid LLRBT. Testing will be the same as project 2 but with an additional “validate” command at the end of the file to check whether or not it is a valid LLRBT. The validate command checks to ensure that:

* The root is black
* Two red nodes are not directly connected
* A node doesn’t have a right red child and a black (or null) left child
* A node doesn’t have a red left child and red left grandchild
* A node doesn’t have a red left and right child