CSE 2320 Lab Assignment 3

Due April 23, 2015

Goals:

1. Understanding of top-down red-black trees.
2. Understanding of tombstones as a simple mechanism for supporting deletions in a data structure.

Requirements:

1. Modify the provided C code (`http://ranger.uta.edu/~weems/NOTES2320/REDBLACKC/`) for maintaining a red-black tree to process a sequence of commands (standard input) from the following list:

   `0` - Exit the program

   `1` *x* - Insert positive integer key *x*, unless *x* is already present in the tree. Besides inserting the key, subtree sizes must be updated. (Processing a duplicate *x* is handled as an update, even though there is no satellite data. Updates may still trigger color flips, which in turn may trigger violations of structural property 3.)

   `2` *x* - Logically delete the item for positive integer key *x* by using a *tombstone*. If there is no item, then the operation is ignored.

   `3` *x* - Find the rank of *x*, i.e. the number of keys in the tree that are smaller than *x* (error message if *x* is not in the tree). Tombstoned items are not included!

   `4` *k* - Find the key with rank *k* (error message if *k* is not legal). Tombstoned items are not included!

   `5` - Print statistics - number of live and dead nodes, along with the number of nodes in the recycling list (which is likely to vary from the provided output).

   `6` - Rebuild tree (unless it has no dead nodes) by collecting live nodes in an ascending order linked list and placing dead nodes on a recycling list (to be used later when inserting). The tree is then rebuilt using an inorder approach *without* calling the insertion code.

   `7` - Print the tree as is done in the sample code, but indicating dead keys with surrounding parentheses.

   `8` - Perform an audit to check the tree for 1) red-black structral properties, 2) inorder traversal property, and 3) correct subtree sizes (number of live nodes in each subtree) to give a final indication that the tree is "clean" or "corrupt".

   Each command must be echoed to standard output. Commands `1`, `2`, `3`, and `4` must be processed in $\Theta(\log n)$ time, where *n* is the total number of nodes. Commands `6` and `8` must be processed in $\Theta(n)$ time. Command `5` must be processed in $\Theta(1)$ time.

2. Email your C source files as *a single zipped file* on Blackboard by 3:15 pm on April 23. One of the comment lines should include the compilation command used on OMEGA.

Getting Started:

1. Command `6` must assign new colors to nodes. One approach is to make the shape of the new tree with *n* live nodes the same as the shape of a binary heap with *n* entries in use.

2. Be sure to observe the special cases that occur for node coloring when there are fewer than two nodes, besides the sentinel, in the tree.

3. You may modify the `tracePrint()` calls that occur, but tracing should be turned off in the code that you submit.

4. A suitable driver is available at `http://ranger.uta.edu/~weems/NOTES2320/LAB/LAB3SPR15/`