Vu Dinh
James Murphree

1. How can we distinguish threads from real child pointers?
   Have Boolean flags to keep track

2. What are the potential advantages of using threaded trees?
   Iterative traversal of tree is possible and more memory-efficient on the stacks

3. Does your implementation use any more memory than the "plain" BST implementation?
   Yes, see (1).
   If yes, can you think of any (obscure?) way to eliminate this extra memory requirement?
   Yes, only one thread is needed, so only 1 boolean is needed. We think we can determine leaf-status without any additional flags. This is 1) not tested, 2) potentially very slow.

4. What is your implementation/test plan.
   Write from Data --> Node --> Tree // test all 3 classes' unit functions
   Implement Threaded build // test in-order traversal
   Implement insert() / remove() // test in various positions
   Implement in-order iterator // test iterator with various tree shapes
   Implement tree re-balancing // test balancing on various tree shapes
   If time permits, implement functional programming

5. What steps must you go through to change the vanilla BST implementation to a TBST?
   While adding child nodes, append threads appropriately with regards to the parent's threads. In detail:
   - If child < parent: thread to parent
   - If child > parent: no thread, right pointer points to NULL
   While removing, beside bumping up one subtree, we have to be careful and deal with existing threads appropriately. Current design proposes that we bump the left subtree up and append the right subtree to the rightmode Node's thread.
   This has the potential to make the tree really tall if certain nodes are removed in succession. We intend to use tree rotation to quickly balance it out a bit.

6. Which BST methods need to be modified/overriden and which will work as is?
   Working as is:
   - begin()
   - end() // these 2 are for the iterator
   - constructors
   - balanceSelf()
   - contains() may remain intact if we wish to search recursively
   - isEmpty()
   - findParentOf()
   Modified:
   - Node::isLeaf() now needs to check for threaded-ness in addition to NULL-ness
   - print() now can use the iterator to abstract traversal details away
   - add() or insert() needs to create threads and modify threaded-ness
   - remove() needs to modify threaded-ness
   - contains() may be modified if we wish to search iteratively

7. What are the advantages and disadvantages of modifying the Carrano BST versus sub-classing from it?
   Modifying:
   ---Advantage:
   ------Able to fix bugs in his code

   ---Disadvantages:
   ------Time consuming
   Subclassing: other way around