4. **(35 points) Balanced Binary Search Trees**

   (a) (2 point) In your own words, list the properties of a **Balanced** Binary Search Tree (BBST). Use terminology discussed in class.

   A BBST is a binary tree that manages to be balanced and spreaded. It does so by updating its nodes' height and balance factor. When one node's balance factor is unexceptable it rotates the the graph so to make it proper again.

   (b) (3 points) What are the respective asymptotic worst-case run-times of each of the following operations of a BBST? Give a Θ bound if appropriate. Do not forget to include the complexity of the rebalancing operation where needed. Justify your answers. You do NOT need to do a line-by-line analysis of code.

   i. Insert O(h) where h is the maximum height of the tree.
   ii. Delete O(h) where h is the maximum height of the tree.
   iii. Find-next O(n) where n is the number of nodes in the tree.
   iv. Find-prev O(n) where n is the number of nodes in the tree.
   v. Find-min O(h) where h is the maximum height of the tree.
   vi. Find-max O(h) where h is the maximum height of the tree.

(c) (15 points) (*You must submit code for this question!*) Submit an implementation of the following **iterative** methods in an AVL Tree. You do not need to submit written answers to the framework from above, but it might be useful for you to consider the answers to those questions when writing code. Note that the `Iter` suffix simply means that the function is iterative. **Keep in mind that an iterative solution cannot make a single recursive call!** You will need to use your implementation of `Node` from the previous question.

1. `insertIter`
2. `deleteIter`
3. `findNextIter`
4. `findPrevIter`
5. `findMinIter`
6. `findMaxIter`

From here, we will prove the efficiency of a balanced binary search tree as it compares to an unbalanced binary search tree using by building trees using a list of integers.

---

5. **(15 points) Constructing Trees**

   (a) (5 points) (*You must submit code for this question!*) Use your **recursive implementation** of your BST and your **iterative implementation** of your AVL Tree from Parts 1 and 2 to construct trees using `getRandomArray(10,000)`. Both trees must be made from the same array. In other words, **do not call the method twice - store the output of the method from `getRandomArray(10,000)` once and use it to construct both trees**.

   (b) (5 points) Did you run into any issues? Test your code on a smaller input (say, `getRandomArray(10)`, and see if you're still running into the same error. If it works on inputs of size 10 but not size 10,000, your code is probably fine and this is expected! Explain why you're running into issues (or might run into issues), using concepts we covered in class.

   In a huge number of nodes the program crashes.

   (c) (5 points) (*You must submit code for this question!*) Use your **iterative implementations** of your AVL Tree and BST from Parts 1 and 2 to construct trees from the input of your implementation of `getRandomArray(10,000)`. Both trees must be made from the same array. In other words, **do not call the method twice - store the output of the method from `getRandomArray(10,000)` once and use it to construct both trees**.

6. **(15 points) Compare Implementations**

   (a) (5 points) (*You must submit code for this question!*) Modify your iterative implementations of your methods in `AVLTree` and `BinarySearchTree` by keeping track of how many times you traverse one level down in the tree. In other words, if you go from a node to its child, add 1 to the counter.

   (b) (5 points) (*You must submit code for this question!*) Construct a BST and AVLTree iteratively using `getRandomArray(10000)`. Compare how many levels we have to traverse in the two trees. You can include a screenshot of your code's output or write it out by hand.

   In contrust with the recursive it does not crash.

   (c) (5 points) (*You must submit code for this question!*) Construct a BST and AVLTree iteratively using `getSortedArray(10000)`. Compare how many levels we have to traverse in the two trees. You can include a screenshot of your code's output or write it out by hand.

7. **(13 points) Extra Credit**

Note: Extra credit problems require significantly more independent research and effort than the for-credit problems. They will also not off-set poor comprehension of previous parts of this project. It is strongly recommended to save these for last.

(a) (3 points) (*You must submit code for extra credit on this question!*) Use time packages in your respective language to quantify (in milliseconds/picoseconds) how much longer it takes to run 10,000 inserts and 10,000 deletes on a Binary Search Tree versus a Balanced Binary Search Tree.

(b) (10 points) *Warning: This problem will be* ***very*** *time consuming!!*
(*You must submit code for extra credit on this question!*) Use https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/ as a guide to learning about Red-Black trees (or R/B Trees), which are another self-balancing tree. Implement a R/B tree that supports the same features as the AVL tree you implemented, and compare run-times in milliseconds.

This concludes the set of problems that must be completed and turned in by 11:59pm on Wednesday, March 4^{th}.

8. **(15 points) Code Review**

   After submitting your code, Sresht will be individually reading and reviewing your code. Comments will be added to your code. At some point before March 12^{th}, it is your responsibility to amend your code and push up a new commit to the same repository. If you sufficiently address all points made in code review, you will receive full credit on this part of the project.

This page is intentionally blank. If you need extra space to answer any questions in this assignment, please use this page to do so. ***Please clearly label what problem and part you are answering if you use this page.***

This page is intentionally blank. If you need extra space to answer any questions in this assignment, please use this page to do so. ***Please clearly label what problem and part you are answering if you use this page.***

This page is intentionally blank. If you need extra space to answer any questions in this assignment, please use this page to do so. ***Please clearly label what problem and part you are answering if you use this page.***

This page is intentionally blank. If you need extra space to answer any questions in this assignment, please use this page to do so. ***Please clearly label what problem and part you are answering if you use this page.***

This page is intentionally blank. If you need extra space to answer any questions in this assignment, please use this page to do so. ***Please clearly label what problem and part you are answering if you use this page.***