CIS 313, Intermediate Data Structures
Fall 2019
CIS 313 Lab 2
Due: November 11, 11:59PM

This lab task is to implement a Binary Search Tree

Overview
Fill out all of the methods in the skeleton code provided. You may add additional methods, but should NOT add additional private variables to the class. You also should not change the name of any of the classes or files. You will implement a working Binary Search Tree in the Tree class in **lab2.py.** Your code will be tested using the test cases in the **test_lab2.py** file. We will be testing your code with additional test cases in Gradescope. So feel free to write as many test cases as possible to thoroughly test your code.

insert
To simplify things, we will not test your binary search tree with duplicate elements.
All insertions will be integer data. Insertion should change one of the leaves of the tree from null to a node holding the inserted value. This should also preserve the ordering requirement that all nodes in the right side of a subtree are greater than the value in the root of that subtree, and all elements in the left side are lesser than the value in the root of the subtree.

delete
When deleting a value, delete the node which contains that value from the tree.
If said node has no children, simply remove it by setting its parent to point to null instead of it.
If said node has one child, delete it by making its parent point to its child.
If said node has two children, then replace it with its successor, and remove the successor from its previous location in the tree.
The successor of a node is the left-most node in the node's right subtree.
If the value specified by delete does not exist in the tree, then the structure is left unchanged.
Note that the delete method returns None. Delete is a complex method to implement. So you need to test your code rigorously to get it right.

find_successor
find_successor takes an int and returns the next highest value in the BST. This method is useful in implementing the delete() method but may also be called on its own.

__find_node
Find takes an int and returns the node in the tree which holds that value. If no such node exists in the tree, return None. This method is private, therefore it can only be used from within the BST class.

contains
Contains is like find_node, but is a public method that only returns True or False rather than the node itself. Contains receives a value and returns true if there is a node with the value in the tree.

min
Returns the node with the minimum value in the tree. Return None if the tree is empty.

max
Returns the node with the minimum value in the tree. Return None if the tree is empty.

Traversals: pre-order, post-order, in-order
You should do this for the pre-order, post-order, and in-order representation of the elements. You will be using generators to implement the methods. You can just modify the __traverse() helper function to implement the traversals. You may use additional helper methods, but all the three traversals can be achieved using a single __traverse method.

Recommended Strategy
First implement the insert function and any one of the traversals (inorder helps in visualizing the tree better). This helps you to get started and check if you are building the tree properly. Create trees which have different shapes. This will also help you debug your other functions. As always write pseudo-code before implementing your methods. Drawing pictures will help immensely when writing your pseudo-code, especially for the delete method. If you start on the last day you will probably not finish. Please don't take this warning as a challenge.

Please refer to the test cases to get a better understanding about using the insert, delete and other functions to build and test your tree. Building these trees and implementing the different methods on these trees is a good way to approach the assignment.

Submission
Compress the lab2.py and test_lab2.py files and upload in Gradescope similar to the previous programming assignments. The test cases file should contain all the additional test cases that you have written to test your code.

Grading
Similar to previous lab assignments, 70 points will awarded in the test cases and 30 points for style. To earn points for style, your code must be clear enough for us to understand . You may not use any data structures from the Python standard library. Some inbuilt functions in python can be used as required. The rubrics for different functions will be provided in Gradescope.