ASSIGNMENT WEEK 3

In class we have seen implementations of several methods for a binary search tree data structure both in Java and Python. For this assignment, you will implement four methods as outlined below in Python.

0. Use the code from class / Canvas as a starting point or start yourself from scratch to have a class **`BST`** with key-value nodes.
1. Implement a ***private*** minimum method (named **`__minimum(self, startNode)`**) based on the pseudocode **`MINIMUM`** from our slides and text. This method will return the ***node*** with minimum key relative to node **`startNode`** (not necessarily **`root`**) if it exists, and **`None`** otherwise.
2. Implement a public minimum method (named **`minimum(self)`**) based on the pseudocode **`MINIMUM`** from our slides and the method **`__minimum()`** you created. This method will return the minimum ***key*** starting from **`root`** if it exists, and **`None`** otherwise. Adjust your code so that the final answer is a key and not a node.
3. Implement a private delete method (**`__delete(self, node, key)`**) which will operate on nodes as our pseudo code does. Within the private function, traverse the tree to find the node with matching key. Then implement the three cases:
    a. If the node does not have a left child, return the right child (possibly **`None`**) and delete the node (mark for garbage collection with **`node=None`**).
    b. If the node does not have a right child, return the left child (possibly **`None`**) and delete the node (mark for garbage collection with **`node=None`**).
    c. If the node has two children, find the successor node using the **`__minimum`** method on the right child. Copy the successor node into the current node and recursively call **`__delete`** on the right child with the successor key.

    When **`__delete`** is executed successfully, return the node.
4. Implement a public delete method (a wrapper method named **`delete(self, key)`**) which will utilize a private delete method (**`__delete(self, node, key)`**) to delete the node with matching key where search starts with node **`node`**.
5. Test and debug your methods. Provide test runs in form of a main (driver) file in which you create appropriate variables, fill the tree with data from the ***provided JSON file***, and run some cases to show your methods work. In particular, show that deleting keys S06E15 and S06E16 will make S06E12 the new maximum key. Do noit forget to upload **`main.py`** and **`bst.py`**!

Each of the problems 1 – 5 will be graded according to the following rubric for a total of 20 points.

SCORE	**4**	**3**	**2**	**1**	**0**
SKILL LEVEL	Response gives evidence of a complete understanding of the problem; is fully developed; is clearly communicated.	Response gives the evidence of a clear understanding of the problem but contains minor errors or is not fully communicated.	Response gives evidence of a reasonable approach but indicates gaps in conceptual understanding. Explanations are incomplete, vague, or muddled.	Response gives some evidence of problem understanding but contains major math or reasoning errors.	No response or response is completely incorrect or irrelevant.