September 19, 2018
CS: DS&A
Proof School

Assignment 3
Due Monday, 9/24

1. Implement a stack using linked lists. This means you should design a `Stack` class, with methods `push(data)` (returning nothing), `pop()` (returning the popped data) and `is_empty()` (returning True/False). Your class should maintain a private attribute of type `LinkedList`, which means that your assignment 3 file should import `linked_list.py`. Use the methods of the LinkedList class (e.g. `insert` and `delete`) when you can.

(Note: A *private attribute* is an internal attribute that shouldn't be referenced by a programmer who's using your class; it's not a part of the "public interface" of the class. Unlike Java, Python doesn't have a way to enforce privateness, but there is a convention you should follow: names of private attributes and methods start with two underscores. So your Stack class should have an attribute named something like `__data_list`.)

2. Implement a queue class using two stacks. This means designing a `Queue` class with methods `enqueue(data)` (returning nothing), `dequeue()` (returning the dequeued data), and `is_empty()` (returning True/False). It should have two private Stack attributes. Give some thought for how to do this! It's not obvious.

3. Implement `delete` for a binary tree. Use the algorithm from class. (Check the class notes if you don't remember it.) Use the binary tree file `binary_tree_for_class.py`. A few notes:

- When you switch nodes, you should switch the actual nodes themselves. In other words, you should *not* copy key and data values. Instead, adjust all the pointers. (The reason for this is that if a user has a pointer to some node, calling `delete` on a different node shouldn't affect their node's key and value.)
- In my code I use `settattr`, which is an awesome Pythonic trick for setting an attribute dynamically. Read up on it! (It's pretty simple.) Its cousin is `getattr`.
- This is... surprisingly tricky. Work it out on paper first!
- You should also use `pretty_print.pyc` so you can print your trees easily and see if your function is working. I've written some testing code for you in the file.

Speaking of `pretty_print.pyc`, I remind you of the Pretty Print Competition. I'm pretty proud of mine, but perhaps you can progressively produce a more pristine prize-worthy pronouncement. Write your own and share with the class! The winner gets gloating rites. (ASCII art only.)