Mod 4 Lab: Linear Data Structures and Linear ADTs

We will implement two abstract data types (stacks and queues) using two different data structures (lists and linked lists).

This lab is extremely easyto cheat on: you can find code for implementing linear ADTs in lectures, the textbook, and any popular coding forum. Do not access these resources during lab. Spend your 75-minutelab period trying your best to implement these classes with your partner; it's the best way to learn this material. Feel free to use these resources after your lab period.

Part 1 - Finish the Linked List

`LinkedList.py` contains:

- complete `Node` class
- almost complete `LinkedList` class.
- test cases for the above

If you run `LinkedList.py`, you should see the `remove_last` test cases fail.

- Modify `remove_last` so that all test cases pass.

Part 2 - Build ADTs with different data structures

`ADT.py` contains skeleton code for list and linked list versions of stacks and queues:

```
from LinkedList import LinkedList

class Stack_L:
    def __init__(self):
        self._L = list()        # Composition: this class has a List

class Stack_LL:
    def __init__(self):
        self._LL = LinkedList() # Composition: this class has a Linked List

class Queue_L:
    def __init__(self): pass

class Queue_LL:
    def __init__(self): pass
```

- Use test driven development (red-green-refactor) to implement the core functionalities for each class:
  - `push` and `pop` for stacks
  - `enqueue` and `dequeue` for the queues
- Use lists (`_L`) or linked lists (`_LL`) as denoted by the class names
- Include tests for each method in the public interface of each class
- You can use `assert` statements or `unittest` - the choice is yours

Submitting

At a minimum, submit the following files:

- `LinkedList.py`
- `ADT.py`

Students must submit to Mimir **individually** by the due date (typically, two days after lab at 11:59 pm EST) to receive credit.

Grading

- 10 - `LinkedList` functionality
- 10 - `Stack_L` functionality
- 10 - `Stack_LL` functionality
- 15 - `Queue_L` functionality
- 15 - `Queue_LL` functionality
- 40 - Tests (manually graded)
    - 10 - `Stack_L`
        - 5 - `push`
        - 5 - `pop`
    - 10 - `Stack_LL`
        - 5 - `push`
        - 5 - `pop`
    - 10 - `Queue_L`
        - 5 - `enqueue`
        - 5 - `dequeue`
    - 10 - `Queue_LL`
        - 5 - `enqueue`
        - 5 - `dequeue`

Feedback

If you have any feedback on this assignment, please leave it here.

We check this feedback regularly. It has resulted in:

- A simplified, clear **Submitting** section on all assignments
- A simplified, clear **Grading** section on all assignments
- Clearer instructions on several assignments (particularly in the recursion module)