Lab 5 - Recursion

Use recursion to implement functionality in the Linked List data structure.

Starter Code

The following methods are implemented for you. Everything but `str` should look familiar. `str` is provided to give you an example of a recursive function applied to a LinkedList.

- `init`
- `len`
- `add_first`
- `remove_first`
- `str`
    - *O*(n^2)
    - returns a string representation of the LL:

```
>>> LL = LinkedList()
>>> for i in range(4): LL.add_first(i)
>>> str(LL)
3-2-1-0
```

    - this is slow! You will probably fail your test cases if you try to use `str()` to implement any funtionality below.

Deliverables

Use recursion to add the attributes described below.

Magic Methods

- `in`
    - *O*(n)
    - returns a boolean describing whether an item is in the LinkedList:

```
>>> LL = LinkedList()
>>> for i in range(4): LL.add_first(i)
>>> 3 in LL
True
>>> 4 in LL
False
```

Non-Magic Methods

- `add_last`
  - *O*(n)
  - adds item to end of LinkedList

```
>>> LL = LinkedList()
>>> for i in range(4): LL.add_last(i)
>>> str(LL)
0-1-2-3
```

Submission

At a minimum, submit the following files:

- `LinkedList.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

- 50 - `in`
  - 25 - uses recursion
  - 25 - functionality
- 50 - `add_last`
  - 25 - uses recursion
  - 25 - functionality

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 (that's this one!))