Module 5 Homework - Recursion

This homework has two independent parts:

1. Use recursion to implement additional functionality in the Linked List data structure.
2. Write a recursive function that converts a decimal number to a different base.

Part 1 - Linked List

Starter Code

The following methods are implemented for you. In addition to the code provided, you have your implementations of `in` and `add_last` from the lab.

- `init`
- `add_first`
- `remove_first`
- `print_all`
    - *O*(n)
    - prints the elements contained in the LL:

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

    - This an example of a non-recursive method calling a recursive helper function, `print_nodes`.

Note that the implementation of `len` has been removed, together with the counter attribute used to keep track of the number of elements. Do not add the counter, as the method will be implemented recursively.

Deliverables

Use recursion to add the attributes described below.

Magic Method

- `len`
    - *O*(n)
    - returns the length of the LinkedList

```
>>> LL = LinkedList()
>>> for i in range(4): LL.add_first(i)
>>> len(LL)
4
```

Non-Magic Methods

- `sum_all`
  - *O*(n)
  - returns the sum of the values contained in the LinkedList

```
>>> LL = LinkedList()
>>> for i in range(4): LL.add_first(i)
>>> LL.sum_all()
6
```

- `deep_copy`
  - *O*(n)
  - returns a new LinkedList containing copies of all the elements

```
>>> LL1 = LinkedList()
>>> for i in range(4): LL1.add_first(i)
>>> LL2 = LL1.deep_copy()
>>> LL1 is LL2
False
>>> isinstance(LL2, LinkedList)
True
```

- `reverse`
  - *O*(n)
  - reverses the LinkedList without copying the data

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

Part 2 - Base Conversion

Write a recursive function that converts an integer given in the decimal system (base 10) to a string representing the integer in a different base (an integer less than 10).

Deliverables

Use recursion to implement the function described below.

- `convert_to_base(number, base)`
  - returns a string representing the number expressed in the given base

```
>>> convert_to_base(593, 3)
210222
```

Note that a recursive helper function is neither required nor recommended in this case.

Submission

At a minimum, submit the following files:

- `LinkedList.py`
- `base_conversion.py`

Students must submit to Mimir **individually** by the due date (typically, the second Wednesday after this module opens at 11:59 pm EST) to receive credit.

Grading

- 20 - `len`
  - 10 - uses recursion
  - 10 - functionality
- 20 - `sum_all`
  - 10 - uses recursion
  - 10 - functionality
- 20 - `deep_copy`
  - 10 - uses recursion
  - 10 - functionality
- 20 - `reverse`
  - 10 - uses recursion
  - 10 - functionality
- 20 - `convert_to_base`
  - 10 - uses recursion
  - 10 - 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!))