Lab 4: Doubly Linked List

Implement an ordered list using a doubly linked list

Before doing this lab, I recommend that you read over section 3.22 and 3.23 in the text. You will be extending the implementation as described in those sections, along with a few modifications specific to this lab – read the descriptions for each method carefully!

The structure of an ordered list is a collection of items where each item holds a relative position that is based upon some underlying characteristic of the item. The ordering is typically either ascending or descending, and we assume that list items have a meaningful comparison operation that is already defined. Many of the ordered list operations are the same as those of the unordered list. To keep things simple, we'll assume our items are integers (could be negative).

Implement the following operations for an **ordered list of integers ordered in ascending order** using a **doubly linked list**. Let the "**head**" of the list be where the "smallest" items are and let the "**tail**" be where the "largest" items are. To keep track of the head and tail of the list, use a single sentinel node as shown in class. You will also need to write thorough test cases for each function.

You may use iterative code for many of the methods, but you **must implement the following methods using recursion**:

- **size()**
- **search()**
- **python_list_reversed()**

For these methods, you should use a helper method that will do the recursion

The following starter files are available on GitHub. Complete the implementations and ensure that your implementations are committed and pushed to GitHub.

- **ordered_list.py**
- **ordered_list_tests.py**