Lab 4: Doubly Linked List

Implement an ordered list using a doubly linked list

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.

Implement the following operations for an **ordered list of items 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 (dummy) node** as shown in class. This means that the head of the list will sentinel.next, and the tail of list will be sentinel.prev. 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

Notes:

- You may assume that all items added to your list can be compared using the > operator, and can be compared for equality/inequality. Make no other assumptions about the items in your list.
- If an attempt is made to add a duplicate item to the list, do not add the item.

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

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