Lab 1: Doubly Linked Ordered 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. You can use my code of doubly linked list provided at my canvas page in syllabus and course material. For conceptual understanding of ordered list use the following link :
https://runestone.academy/ns/books/published/pythonds/BasicDS/ImplementinganOrderedList.html#analysis-of-linked-lists. 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. This means that any objects added to the list must have an _lt___method and an _eq___method. Make **no other assumptions** about the items in your list.

The following starter files are available.

- **ordered_list.py**

```
from dataclasses import dataclass

 @dataclass
 class Node:
   value: int
   prev_node: None
   next_node: None

 @dataclass
  class doubly_Ordered_List:



  '''A doubly-linked ordered list of items, from lowest (head of list) to highest (tail of
  list)'''

      def is_empty(self):
```

```
        '''Returns True if OrderedList is empty
            MUST have O(1) performance'''
        pass

    def add(self, item):
        '''Adds an item to OrderedList, in the proper location based on ordering of items
           from lowest (at head of list) to highest (at tail of list) and returns True.
           If the item is already in the list, do not add it again and return False.
           MUST have O(n) average-case performance'''
        pass

    def remove(self, item):
        '''Removes the first occurrence of an item from OrderedList. If item is removed (was
in the list)
          returns True.  If item was not removed (was not in the list) returns False
           MUST have O(n) average-case performance'''
        pass

    def index(self, item):
        '''Returns index of the first occurrence of an item in OrderedList (assuming head of
list is index 0).
           If item is not in list, return None
           MUST have O(n) average-case performance'''
        pass

    def pop(self, index):
        '''Removes and returns item at index (assuming head of list is index 0).
           If index is negative or >= size of list, raises IndexError
           MUST have O(n) average-case performance'''
        pass

    def search(self, item):
        '''Searches OrderedList for item, returns True if item is in list, False otherwise"
           To practice recursion, this method must call a RECURSIVE method that
           will search the list
           MUST have O(n) average-case performance'''
        pass

    def python_list(self):
        '''Return a Python list representation of OrderedList, from head to tail
           For example, list with integers 1, 2, and 3 would return [1, 2, 3]
           MUST have O(n) performance'''
        pass

    def python_list_reversed(self):
        '''Return a Python list representation of OrderedList, from tail to head, using
recursion
           For example, list with integers 1, 2, and 3 would return [3, 2, 1]
```

```
            To practice recursion, this method must call a RECURSIVE method that
            will return a reversed list
            MUST have O(n) performance'''
        pass

    def size(self):
        '''Returns number of items in the OrderedList
            To practice recursion, this method must call a RECURSIVE method that
            will count and return the number of items in the list
            MUST have O(n) performance'''
        pass
```

- **ordered_list_tests.py**

```
Import unittest
from ordered_list import *

class TestLab4(unittest.TestCase):

    def test_simple(self):
        t_list = OrderedList()
        t_list.add(10)
        self.assertEqual(t_list.python_list(), [10])
        self.assertEqual(t_list.size(), 1)
        self.assertEqual(t_list.index(10), 0)
        self.assertTrue(t_list.search(10))
        self.assertFalse(t_list.is_empty())
        self.assertEqual(t_list.python_list_reversed(), [10])
        self.assertTrue(t_list.remove(10))
        t_list.add(10)
        self.assertEqual(t_list.pop(0), 10)


if __name___== '__main__':
    unittest.main()
```