Lab4: Linked List

Due: 10/12 @ 11PM

Implement an ordered list using doubly linked list

This lab will be based on group of two. Peek your partner and give me the members name during lab hour.

Before doing this lab make sure to read over section 3.22, 3.23 carefully. You will be extending the implementation as described in the text.

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 integers ordered in ascending order** using a **doubly linked list**. The "**head**" of the list be where the "smallest items are and let "**tail**" be where the largest items are. You may use whatever mechanism you like to keep track of the head and tail of the list. E.g. references or sentinel nodes.

- OrderedList () creates a new ordered list that is empty. It needs no parameters and returns an empty list.
- add (item) adds a new item to the list making sure that the order is preserved. It needs the item and returns nothing. Assume the item is not already in the list.
- remove (item) removes the item from the list. It needs the item and modifies the list. Return the position of removed item if it is in the list, otherwise return -1 (as not found).
- search_forward (item) searches for the item in the list. It needs the item and returns the boolean value True if the item is in the list and False if the item is not in the list.
- **search_backward** (item) searches for the item in the list starting from the tail of the list. It needs the item and returns the boolean value True if the item is in the list and False if the item is not in the list.
- is_empty () tests to see whether the list is empty. It needs no parameters and returns a boolean value. True if the list is empty and False if any items are in the list.
- size () returns the number of items in the list. It needs no parameters and returns an integer.
- index (item) returns the position of item in the list. It needs the item and returns the index. If it is not in the item it returns -1(as not found).
- pop () removes and returns the last item in the list. It needs nothing and returns an item.
- pop (pos) removes and returns the item at position pos. It needs the position and returns the item. If it is not in the item it returns -1(as not found). **pop(pos) should compare pos to the size of the list and search from the head if pos <= size/2 and from the rear if pos > size/2.**

CPE 202 **Lab 4** **Fall 2017**

Submit two files to PolyLearn

1. ordered_list.py
2. ordered_list_tests.py

You can submit one version for each group. Divide the job between each member.

For documentation use signature of the design recipe for each function (input-output and purpose statement)

For your classes use data definition of design recipe.

Just write __rper__ for your classes as boilerplate.

No need for templet.