Lab #8
Due Date: 10/12/2018, 11:59PM

Instructions:
- The work in this lab must be completed alone and must be your own. Do not copy code from online sources. That is considered plagiarism.
- Use the starter code provided on this CANVAS assignment. Do not change the function names or given started code on your script
- The file name must be LAB8.py
- **A doctest is provided as an example of code functionality. Getting the same result as the doctest does not guarantee full credit. You are responsible for testing your code with enough data as possible.**
- Each function must return the output (Do not use print in your final submission otherwise, you will get a -1 pt deduction)
- Do not include test code outside any function in the upload. Remove all your testing code before uploading your file. Do not include the input() function in your submission.

Assignment Project Exam Help

Goal:
In class, we have been working on the implementation of a singly linked list. That data structure keeps the elements of the linked list unsorted. Based on the LinkedList code, implement the data structure OrderedLinkedList with the following characteristics:

- *OrderedLinkedList*() creates a new ordered list that is empty. It needs no parameters and returns nothing. You can assume the items in the list are **unique**
- *add*(*item*) adds a new Node with value=item to the list making sure that the ascending order is preserved. It needs the item and returns nothing.
- *pop*() removes and returns the last Node in the list. It needs nothing and returns the **value** of the Node
- *isEmpty*() tests to see whether the list is empty. It needs no parameters and returns a boolean value.
- *len*(*list_object*) returns the number of items in the list. It needs no parameters and returns an integer.

NOTE: To grade this assignment, the grading script will perform a series of mixed linked list operations and compare the final status of your list. Verify that all your methods work correctly when mixed together.

EXAMPLE:
```
>>> x=OrderedLinkedList()
>>> x.isEmpty()
True
>>> x.pop()
'List is empty'
>>> x.add(8)
>>> x.add(7)
>>> x.add(3)
```

```
>>> x.add(-6)
>>> x.add(58)
>>> x.add(33)
>>> x.add(1)
>>> x.add(-88)
>>> print(x)
Head:Node(-88)
Tail:Node(58)
List:-88 -6 1 3 7 8 33 58
>>> x.isEmpty()
False
>>> len(x)
8
>>> x.pop()
58
>>> print(x)
Head:Node(-88)
Tail:Node(33)
List:-88 -6 1 3 7 8 33
>>> x.pop()
33
>>> x.pop()
8
>>> x.pop
7
>>> print(x)
Head:Node(-88)
Tail:Node(3)
List:-88 -6 1 3
>>> len(x)
4
>>> x.add(-4)
>>> print(x)
Head:Node(-88)
Tail:Node(3)
List:-88 -6 -4 1 3
>>> x.add(8)
>>> x.pop()
8
>>> print(x)
Head:Node(-88)
Tail:Node(3)
List:-88 -6 -4 1 3
```

Tips:
  – Work on paper first to see what pointers need to be updated based on the position of the Node
  – Make sure you update the head and tail pointers according to the operation performed
  – Starter code contains the special methods __str__ and __repr__, use them to ensure your methods are updating the elements in the list correctly

Deliverables:
  • Submit your code in a file name LAB8.py to the Lab8 CANVAS assignment before the due date