CMPSC-132: Programming and Computation II
Spring 2020

Lab #4

Due Date: 03/06/2020, 11:59PM
10 pts + Extra Credit

Read the instructions carefully before starting the assignment. Make sure your code follows the stated guidelines to ensure full credit for your work.

Instructions:

- The work in this lab must be completed alone and must be your own.
- **Download the starter code file from the LAB4 Assignment on Canvas. Do not change the function names on your script.**
- 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 debugging and testing your code with enough data, you can share ideas and testing code during your recitation class. As a reminder, Gradescope should not be used to debug and test code!
- Each function must return the output (Do not use print in your final submission, otherwise your submissions will receive a -1 point deduction)
- **Do not include test code outside any function in the upload. Printing unwanted or ill-formatted data to output will cause the test cases to fail. Remove all your testing code before uploading your file (You can also remove the doctest). Do not include the input() function in your submission.**

Goal:

[10 pts] During the in-person Linked List Hands On session (February 28), we worked 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 SortedLinkedList with the following characteristics:

- *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. The items in the list might not be unique, but you can assume every value will be numerical (int and float)
- *pop*() removes and returns the last Node in the list. It needs nothing and returns the **value** of the Node
- *len()* returns an integer that represents the number of Nodes in the list
- *isEmpty()* returns True is the list is empty, False otherwise

- You are not allowed to swap data from the nodes when adding the node to be sorted or to use any other data structures to copy the elements of the Link List for sorting/manipulating purposes. Traversing the linked list and updating pointers are the only required and acceptable operations
- You are not allowed to modify the given constructor
- You are not allowed to use any kind of built-in sorting method or Linked List tools

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__. Do not modify them, use them to ensure your methods are updating the elements in the list correctly

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=SortedLinkedList()
>>> x.pop()
>>> x.add(8.76)
>>> 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 8.76 33 58
>>> x.pop()
58
>>> print(x)
Head:Node(-88)
Tail:Node(33)
List:-88 -6 1 3 7 8.76 33
>>> x.pop()
33
>>> x.pop()
8.76
>>> x.pop()
7
>>> print(x)
Head:Node(-88)
Tail:Node(3)
List:-88 -6 1 3
>>> x.add(-4)
>>> x.add(-4)
>>> x.add(2)
>>> x.add(2)
>>> x.add(3)
>>> print(x)
Head:Node(-88)
Tail:Node(3)
List:-88 -6 -4 -4 1 2 2 3 3
>>> x.add(65)
>>> x.add(-100)
>>> x
Head:Node(-100)
Tail:Node(65)
List:-100 -88 -6 -4 -4 1 2 2 3 3 65
>>> x.add(85)
```

```
>>> x.add(85)
>>> x
Head:Node(-100)
Tail:Node(85)
List:-100 -88 -6 -4 -4 1 2 2 3 3 65 85 85
```

EXTRA CREDIT: +2 pts regardless of the 10-score maximum

In your SortedLinkedList class write the code to implement the method *replicate*. It takes no parameters and returns a new SortedLinkedList object where each element of the linked list appears value_of_the_node number of times. Negative numbers and floats are ignored, they are not repeated but should be included in the list. For 0, the node is added in the new list only once. Method returns None if the list is empty.

```
>>> x=SortedLinkedList()
>>> x.add(4)
>>> x.replicate()
Head:Node(4)
Tail:Node(4)
List:4 4 4 4
>>> x.add(-23)
>>> x.add(2)
>>> x.add(1)
>>> x.add(20.8)
>>> x
Head:Node(-23)
Tail:Node(20.8)
List:-23 1 2 4 20.8
>>> x.replicate()
Head:Node(-23)
Tail:Node(20.8)
List:-23 1 2 2 4 4 4 4 20.8
>>> x.add(-1)
>>> x.add(0)
>>> x.add(3)
>>> x.replicate()
Head:Node(-23)
Tail:Node(20.8)
List:-23 -1 0 1 2 2 3 3 3 4 4 4 4 20.8
>>> x
Head:Node(-23)
Tail:Node(20.8)
List:-23 -1 0 1 2 3 4 20.8
>>> x.add(2)
>>> x.replicate()
Head:Node(-23)
Tail:Node(20.8)
List:-23 -1 0 1 2 2 2 2 3 3 3 4 4 4 4 20.8
```

Deliverables:

- Submit your code in a file name LAB4.py to the Lab4 Gradescope assignment before the due date