CMPSC-122: Intermediate Programming
Spring 2018

Lab #9

Due Date: 03/23/2018, 11:59PM

Instructions:

- The work in this lab must be completed alone.
- If you need guidance, attend to your recitation class.
- Read the "Submitting assignments to Vocareum" file for instructions on how to submit this lab
- Do not change the function names or given code on your script
- The file name must be LAB9.py (incorrect name files will get a 0 score)
- You are responsible for testing your code. Use `python -i LAB9.py` in your terminal (or command prompt) to provide input to your functions. Test with as many data as you feel comfortable
- Each function must return the output (Do not use print in your final submission)
- Do not include test code outside any function in the upload. Remove all your testing code before uploading your file. If you are using input() to insert values in your functions and print to see the values, remove them.

Exercise 1 [10 pts]

NOTE: There is no partial credit for this exercise, you must ensure the entire code works properly after the addition of new methods.

In class, we worked on the implementation of a linked list with the following characteristics:

- *LinkedList*() creates a new list that is empty. It needs no parameters. We are assuming items in the list are **unique**
- *add*(item) adds a new Node with value=item to the beginning of the list. It needs the item and returns nothing.
- *remove*(item) removes the Node with value=item from the list. It needs the item and modifies the list.
- *search*(item) searches for the Node with value=item in the list. It needs the item and returns a boolean value.
- *isEmpty*() tests to see whether the list is empty. It needs no parameters and returns a boolean value.
- *size*() returns the number of items in the list. It needs no parameters and returns an integer.
- *append*(item) adds a new Node with value=item to the end of the list. It needs the item and returns nothing.

Modify the LinkedList class to allow the following linked list operations:

- *insert*(after,item) adds a new Node with value=item to the list after the Node with value=after. It needs the value of the existing node and the new value to be inserted, returns nothing. [5 pts]
- *pop*() removes and **returns** the value of the last Node in the list. It needs no parameters and modifies the list and returns an item. [5pts]

EXAMPLE

```
>> linked_list = LinkedList()
>>> linked_list.add(9)
9
>>> linked_list.add(8)
8 9
>>> linked_list.add(7)
7 8 9
>>> linked_list.add(6)
6 7 8 9
>>> linked_list.add(5)
5 6 7 8 9
>>> linked_list.insert(6,1)
>>> linked_list.printList()
5 6 1 7 8 9
>>> linked_list.insert(8,4)
>>> linked_list.printList()
5 6 1 7 8 4 9
>>> linked_list.pop()
9
>>> linked_list.printList()
5 6 1 7 8 4
>>> linked_list.append(15)
>>> linked_list.printList()
5 6 1 7 8 4 15
>>> linked_list.size()
7
>>> linked_list.remove(7)
>>> linked_list.printList()
5 6 1 8 4 15
>>> linked_list.pop()
15
>>> linked_list.printList()
5 6 1 8 4
>>> linked_list.append(63)
>>> linked_list.printList()
5 6 1 8 4 63
>>> linked_list.isEmpty()
False
```

Tips:

- Remember that the insert method takes values, not nodes
- You can modify the search method to return the Node and use that output to implement the insert method.