CMPSC-122: Intermediate Programming
Spring 2018

Homework 7

Due Date: 04/18/2018, 7:59 am
100 points

Overview

A data structure is a particular way of organizing and storing data in a computer so that it can be accessed and modified efficiently. More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data. [1]

Instructions:

In class, we discussed the Doubly Linked List Abstract Data Type (section 1.1.4 of your Lecture Notes). A Doubly Linked List contains an extra pointer (known as previous pointer) allowing the traversal of the list in both forward and backward direction. When implementing the LinkedList data structure, we observed the importance of updating the pointers in order to get access to the entire list. Implement the data structure DoublyLinkedList with the following characteristics:

- *DoublyLinkedList*() creates a new doubly linked list that is empty. It needs no parameters and returns nothing. Assume the items in the list are **unique**
- *addFirt*(*item*) adds a new Node with value=item at the beginning of the list. It needs the item and returns nothing. [20 pts]
- *addLast*(*item*) adds a new Node with value=item at the end of the list. It needs the item and returns nothing. [20 pts]
- *addBefore*(*pnode_value, item*) adds a new Node with value=item before the Node with value=pnode_value. It needs the value of the reference Node and the item to be added, returns nothing. You can assume the reference node is in the list. [30 pts]
- *addAfter*(*pnode_value, item*) adds a new Node with value=item after the Node with value=pnode_value. It needs the value of the reference Node and the item to be added, returns nothing. You can assume the reference node is in the list. [30 pts]

NOTE: There is no partial credit for methods that do not work properly. Code will be tested calling all methods and comparing the final list

EXAMPLE:

```
>>> dll=DoublyLinkedList()
>>> dll.addFirst(5)
>>> dll.addFirst(9)
>>> dll.addFirst(4)
>>> dll.addFirst(3)
>>> dll.head
3
>>> dll.addLast(8)
>>> dll.addLast(12)
```

[1] https://en.wikipedia.org/wiki/Data_structure

```
>>> dll.addLast(56)
>>> dll.printDLL()
Traversal Head to Tail
3 4 9 5 8 12 56
Traversal Tail to Head
56 12 8 5 9 4 3
>>> dll.addBefore(56,44)
>>> dll.addBefore(9,32)
>>> dll.addAfter(56,756)
>>> dll.addAfter(8,47)
>>> dll.printDLL()
Traversal Head to Tail
3 4 32 9 5 8 47 12 44 56 756
Traversal Tail to Head
756 56 44 12 47 8 5 9 32 4 3
>>> dll.addLast(999)
>>> dll.addFirst(856)
>>> dll.head
856
>>> dll.printDLL()
Traversal Head to Tail
856 3 4 32 9 5 8 47 12 44 56 756 999
Traversal Tail to Head
999 756 56 44 12 47 8 5 9 32 4 3 856
```

Hints:

- The method getNode(item) returns the Node with value=item
- The method printDLL() will traverse the list in both directions. If your pointers were correctly updated, you should be able to see all the elements of the list in both traversals.

Deliverables:

- Upload the file HW7.py to the Homework 7 Vocareum assignment

[1] https://en.wikipedia.org/wiki/Data_structure