CSC 242
Exercise 4

Complete the **LinkedListIterator** for the **LinkedList** implementation of a list. As discussed (Ch. 9, p. 275), the **insert**, **remove**, and **replace** methods will run in constant time.

Use the **testlist.py** file to test the iterator and verify that exceptions are raised when preconditions are violated. Also, test the removal of all items in the **LinkedList** using both of the following techniques:

```
    print("Removing all items (reverse): Expect []: ", end = "")
    listIterator.last()
    while listIterator.hasPrevious():
            listIterator.previous()
            listIterator.remove()
    print(lyst)
    print("Length:", len(lyst))
```

and

```
    print("Removing all items (forward): Expect []: ", end = "")
    listIterator.first()
    while listIterator.hasNext():
            listIterator.next()
            listIterator.remove()
    print(lyst)
    print("Length:", len(lyst))
```

Before beginning this exercise, answer the following questions:

1) Given the following line of code found in the **next** and **previous** method of the **LinkedListIterator** class: "`if self._modCount != self._backingStore.getModCount():`", identify the scenario when it will evaluate to true?

2) Explain what the following statement does in the **AbstractCollection**'s **__str__** method call: `", ".join(map(str, self))`.

3) Create some diagrams similar to the **LinkedList** shown in **Figure 1** that exhibits the following code from the **insert** method (step by step—be sure to label **theNode** and **newNode**):

```
theNode = self._getNode(i)
newNode = TwoWayNode(item, theNode.previous, theNode)
theNode.previous.next = newNode
theNode.previous = newNode
```

inserting the item 'Omega' at the end of the **LinkedList** shown in **Figure 1** using:

```
mylist.insert(len(mylist), 'Omega')
```

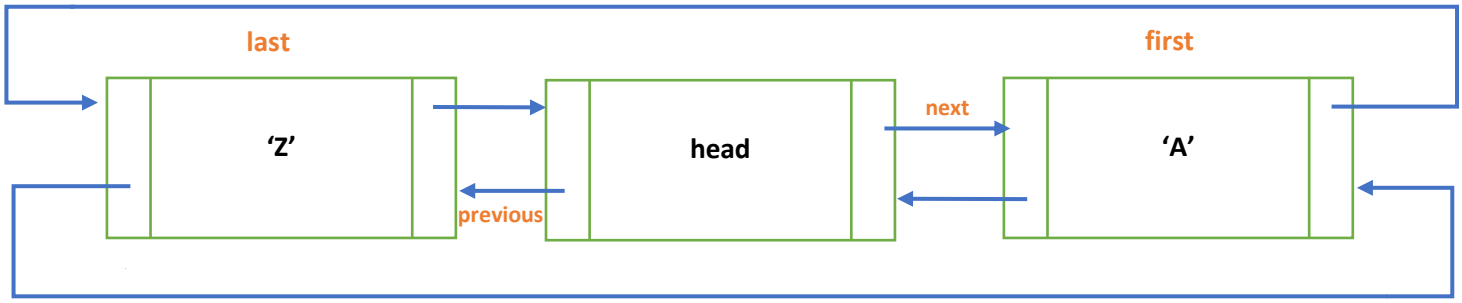

Figure 1