**CMPSC 112 – Introduction to Computer Science II (Spring 2015)**
**Prof. John Wenskovitch**
`http://cs.allegheny.edu/~jwenskovitch/teaching/CMPSC112`

**Lab 4 - Implementing a Doubly-Linked List**
**Due (via Bitbucket and hard copy) Monday, 23 February 2015**
**30 points**

## Lab Goals

- Practice working with linked lists
- Update our recent lecture code to turn our `SinglyLinkedList.java` data structure into a doubly linked list
- Answer a few questions to test your knowledge about the class content to-date

## Assignment Details

In this lab, we will continue our discussion of linked lists from the past few classes. We will alter our `SinglyLinkedList` implementation to create a doubly linked list. You will also answer a few short questions at the end.

Please note that this is a two-week lab due to Exam 1.

### Doubly Linked List (20 points)

In the past few classes, we have been implementing a singly linked list data structure. In a singly linked list, each node in the list contains a reference to the list item that follows it. We also briefly introduced the idea of a doubly-linked list, in which each node in the list contains a reference to the list item that follows it, as well as a reference to the list item that precedes it.

In this lab, we will update our singly linked list code into a new class called `DoublyLinkedList.java`. Much of the functionality will remain the same, but you will need to keep track of the additional reference through insertions and deletions to the list.

In particular, you should implement each of the following functions:

1. In your `Node` subclass, you should include a new instance variable called `previous` of type `Node`, and also implement the related accessor `getPrevious()` and mutator `setPrevious()` functions. You will also need to update the `Node` constructor to take a `previousNode` parameter in addition to the existing `newValue` and `nextNode` parameters. Finally, your `Node` subclass will need a `hasPrevious()` function that mimics the functionality of the existing `hasNext()` function.

2. In your `add()` function, you will need to add a reference to the old `tail` when creating the `Node` you are inserting.
3. In your `remove()` function, you will need to update two references: set the `next` value of the `previous Node` to the `next Node`, and set the `previous` value of the `next Node` to the `previous Node`.
4. Your `get()` function should not need to be changed from the `SinglyLinkedList` version. Therefore, you will add a new `getFromEnd()` function that mimics the same behavior, but starts from the end and uses the previous links to iterate through the list.

A framework has been provided for you in the `lab4/src` directory. This framework includes a function called `testPreviousLinks()`. Please do not modify this function – it is used to test that all of your `previous` references are correct.

Also in the `lab4/src` directory is a test file called `DoublyLinkedListTester.java`. Your `DoublyLinkedList` class should work with this test file when you are finished. The output of this `DoublyLinkedListTester` file should appear as follows:

```
Printing list contents:
1
2
3
4

Printing list contents:
1
2
4

1

true

Emptying list
true

Printing list contents:
17
```

**Additional Questions (10 points)**

Please answer the following questions thoroughly:

1. Give an algorithm for finding the second-to-last node in a singly linked list.
2. Give an algorithm for concatenating two singly linked lists $L$ and $M$ into a single list $L'$, which contains all of the nodes of $L$ followed by all of the nodes of $M$.
3. Suppose you are given two circularly linked lists, $J$ and $K$. Give an algorithm for determining if $J$ and $K$ store the same sequence of elements (but perhaps with different starting points).

## Submission Details

For this assignment, your submission (to both your BitBucket repository and by hardcopy) should include the following:

1. Your source code for `DoublyLinkedList.java`
2. Sample output showing that your class works with `DoublyLinkedListTester.java`
3. The answers to the questions from the "Additional Questions" section
4. An Assignment Information Sheet for your two code files

Please note that each student in the class is responsible for completing and submitting their own version of this assignment. However, you also will be assigned to work to a team that is tasked with ensuring that all of its members are able to complete each step of the assignment. Team members should make themselves available to each other to answer questions and resolve any problems that develop during the laboratory session. While it is acceptable for members of a team to have high-level conversations, you should not share source code or full command lines with your team members. To ensure that you can communicate effectively, members of each team should sit next to each other in the room. Please see the instructor if you have questions about this policy.