# *Programming III (420-B31-HR)*

# *Lab 5 –Doubly Linked Lists*

Date assigned & due: Tuesday, September 20, 2016

**Objectives:**

1. Learn to:
2. add and delete nodes in a doubly linked data structure.
3. traverse a doubly linked list forwards and in reverse.

**Reference:**

PowerPoint Slides from Moodle: *Doubly\_Linked\_Lists.pptx*

**To be handed in:**

1. Your ***username*\_B31\_L05\_Doubly\_Linked\_Lists** folder should be zipped and uploaded to **Moodle**. Show the teacher your lab before you leave for marking.
2. The **Lab 5 Review Quiz** on **Moodle** should be completed.

# Lab 4 Completion

If you did not complete lab 4, then go back and finish part g, h and i in the Singly Linked List section. Submit your updated lab 4 under the lab 5 Moodle submission. (You will end up with a lab 4 and lab 5 submission under Lab 5.)

# Doubly Linked Lists

**Objectives:** Learn to add and delete nodes in a doubly linked data structure and to traverse it both forwards and backwards.

**To Start:**

1. Copy the **B31\_L05\_ DoublyLinkedLists** folder from the **Moodle page** for the course. Rename it to ***username*\_B31\_L05\_ DoublyLinkedLists**.
2. Start **Eclipse** and use your **420-B31\Labs** folder as your workspace.
3. Create a new **Java Project** called ***username*\_B31\_L05\_ DoublyLinkedLists**.

**To Do:**

## Code the private **findLastOccurrence (E target)** method for the **DoublyLinkedList** class. The method should return the last node in the list that contains the target. If the target is not in the list, the method should return null. The pseudocode for the method is as follows:

Set cursor to tail’s predecessor

While the cursor is not at the head node

If the element field of the cursor node matches the target element

Return a reference to cursor node // success

Else

Move cursor to its predecessor // follow the predecessor link

Return null // indicate failure

## Code the public **countOccurrences (E target)** method for the **DoublyLinkedList** class. The method should return the number of nodes in the list that contain the target.

## Code the public **deleteLastOccurrence(E target)** method for the **DoublyLinkedList** class. The method should delete the last occurrence of the target in the list and return the target if the delete was successful. It should return null if the target was not in the list.

## Run the **DoublyLinkedListTest** jUnit test case to test your **findLastOccurrence()**, **countOccurrences()** and **deleteLastOccurrence()** methods. The test cases are shown in the appendix. If the **DoublyLinkedListTest** jUnit file has syntax errors when you open it, then you need to add Junit to your build path. (**Project -> Properties -> Java Build Path and add Junit 4**.)

## Create a method in **DoublyLinkedList** to add an element to the end of the list. Call the method **addAtEnd(E target)**.

## Create a driver program called **ListTraverser** that creates a **DoublyLinkedList** list of the seven dwarfs from Snow White. Use your **addAtEnd()** method to add each dwarf to the end of the list. Print the list in forward order and then in reverse order. The output should look like:

Going forward the 7 dwarfs are:

Sneezy

Sleepy

Dopey

Doc

Happy

Bashful

Grumpy

Going backwards the 7 dwarfs are:

Grumpy

Bashful

Happy

Doc

Dopey

Sleepy

Sneezy

# Review

**Objectives:** Review terminology and linked lists.

## Log on to **Moodle**, go to the **Programming III** course page and complete the **Lab 5 Review Quiz**.

**Marking Scheme**

|  |  |
| --- | --- |
| **Question** | **Out of** |
| Singly Linked Lists - from Lab 4 |  |
| CSProgram reviseProgram() | 19 |
| SinglyLinkedList replace() | 9 |
| reviseProgram() junit test | 5 |
| DoublyLinkedList   * findLastOccurrence() | 9 |
| * countOccurrences() | 8 |
| * deleteLastOccurrence() | 13 |
| * addAtEnd() | 3 |
| ListTraverser | 12 |
| **Total** | **78** |