# Data Structures Assignment

Subversion (SVN) Repository URL: <https://semi-r123-w032:8443/svn/cp12-sept19>

|  |  |
| --- | --- |
| Everyone Must Complete | **SimpleLinkedList** Ordered with the elements from newest to oldest, or oldest to newest. (Either is fine, so long as you are consistent.)  The test cases I am providing will test for consistency!  Allows Duplicates |

## Then you may decide on your difficulty level:

You will hand in either Level 1 or Level 2. Not both.

|  |  |  |
| --- | --- | --- |
| Difficulty | Level 1 (Moderate Difficulty) | Level 2 (Harder Difficulty) |
| Classes You Need To Complete | **SortedLinkedSet**  * The List keeps a reference to the head of the list. * Remain in sorted order from smallest to largest * Rejects Duplicates  PriorityStack  * The Stack keeps a reference to the Top of the Stack (highest priority item) * When you add an item with a priority level, it is inserted at that level. * If there is a tie in priority, the element added first must be higher in the stack. * When you add an item without a priority level, it is added as a tie with the currently lowest priority item. If there are no items, give it priority 1. * When you “Pop” the stack, it gives back the highest priority item it has. * When you add an item that is already in the stack, you change the priority from the old priority to the new one. **Note**: You must re-use the old StackNode, not make a new one! | **BinaryTree**   * Rejects Duplicates * Starts at the top of a tree at the Root node. * Each TreeNode<T> and has a left and a right * As you proceed to the left, values are smaller than the current element. To the right, values are larger. |
| Test Suite | Level1Tests.java | Level2Tests.java |
| Marks | Max 86% | Max 100% |
| To Get 86% | All test cases pass | Removing a Node with 0/1 child(ren) works properly  Removing Root Node with 0/1 child(ren) works properly |
| To Get 100% | Go do Level 2 instead | Removing a Node with 2 children works properly  Depth & Breadth First Traversal passes |

## How these test cases work:

They rely extensively on add() and size(). I will suggest beginning with this order:

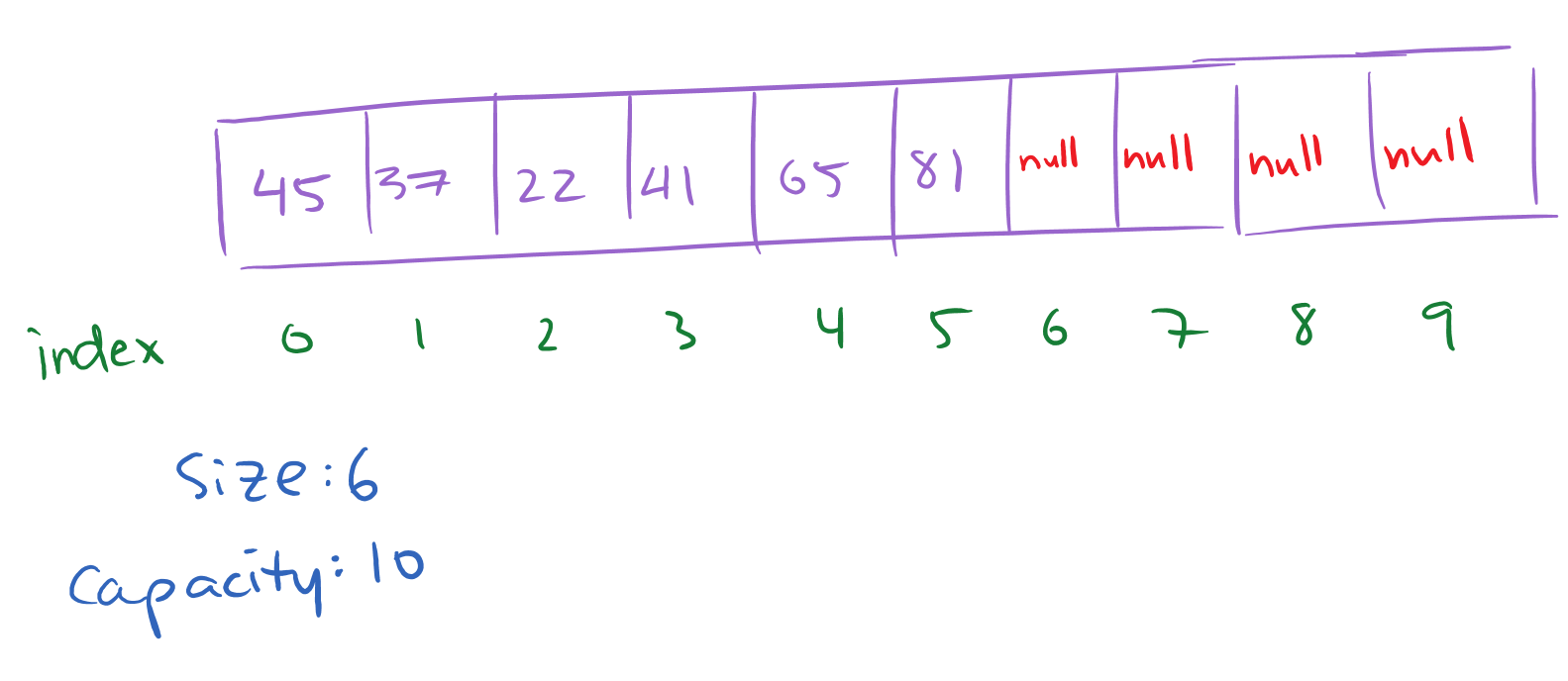
1. add() and size()
2. contains()
3. isEmpty()
4. clear()

## I reserve the right to:

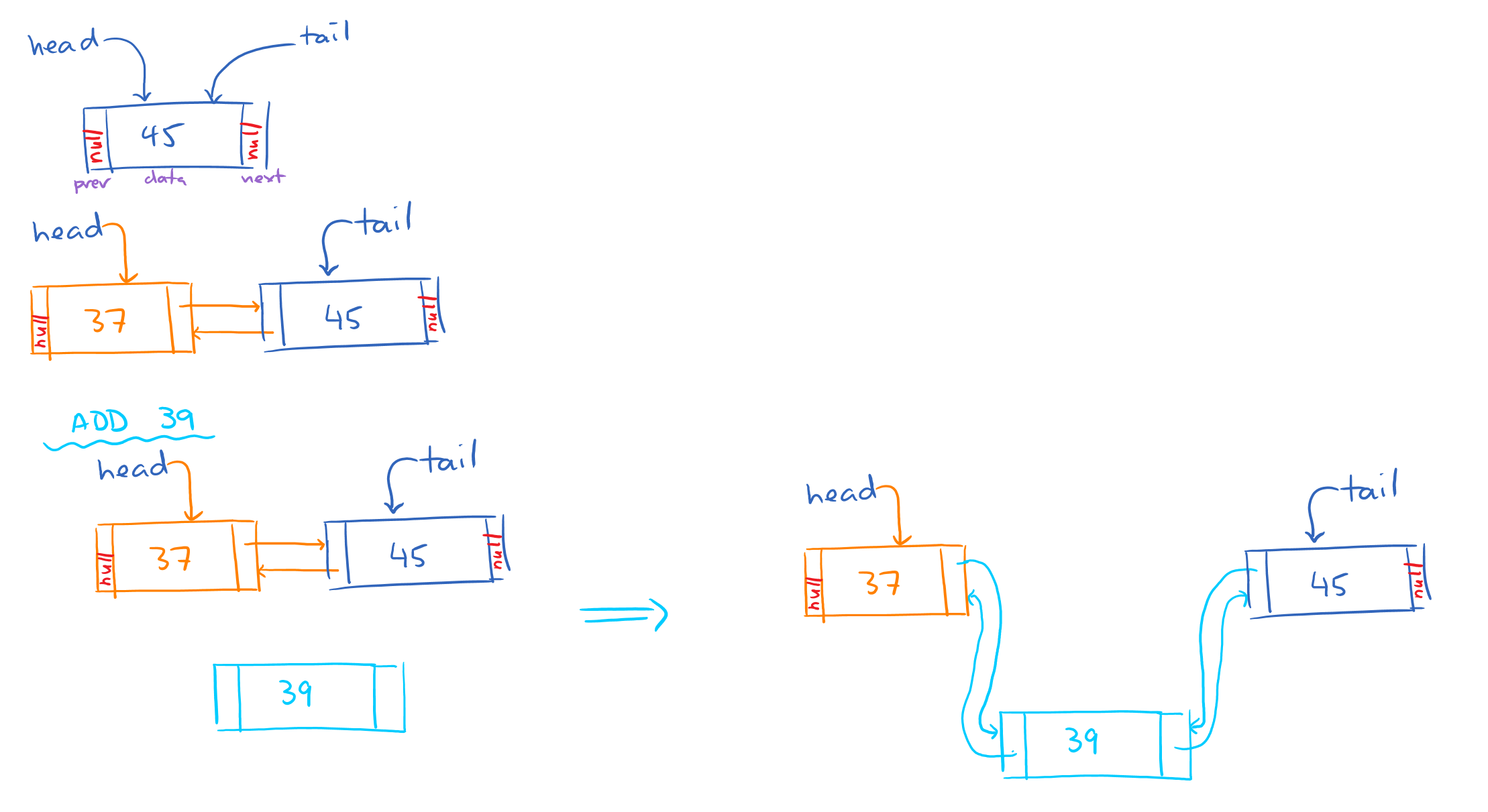
* add/subtract learning outcomes from this list to balance the difficulty level as we proceed through this assignment.
* add/subtract test cases from the SVN server. You are responsible for synchronizing your tests project at the start of every class to make sure you have the most up-to-date version.

# Example Diagrams:

## ArraySet



## Double Linked List



## Binary Tree

