**COMP 132 - Homework # 12**

**Abstract Data Types**

1. Consider the following lines of code which create several LinkedNode objects (see LinkedNode.java on the homework webpage):

String o0 = "Red";

String o1 = "Green";

String o2 = "Blue";

String o3 = "Yellow";

LinkedNode sln0 = **new** LinkedNode(o0);

LinkedNode sln1 = **new** LinkedNode(o1);

LinkedNode sln2 = **new** LinkedNode(o2);

LinkedNode sln3 = **new** LinkedNode(o3);

Draw the linked list that would be produced by the following snippets of code:

a. sln1.next = sln3;

sln2.next = sln0;

sln3.next = sln2;

b. sln0.next = sln3;

sln2.next = sln3;

sln3.next = sln1;

2. Give the implementation of each of the following methods as they would appear if you were to add them to the LinkedList132 class available on the homework webpage. Implement each method by directly manipulating the list (i.e. don’t call the getNode, insertNode or removeNode methods). The main method of LinkedList132 demonstrates an approach for testing your implementations.

a. countNodesAfter(LinkedNode node): return the number of nodes that come after node in the list.

b. swap(LinkedNode pred1, LinkedNode pred2): Swap the node following pred1 with the node following pred2. Move the actual nodes by manipulating their next references, do not simply swap the elements contained in the nodes. You may assume that there is at least one node between pred1 and pred2. You may also assume that there are at least two nodes following pred2.

c. jumpByN(LinkedNode pred, int n): Move the node following pred later in the list by jumping it over n successive nodes. Move the actual nodes by manipulating their next references, do not simply move the elements contained in the nodes. For example, if pred refers to the node at index 2 in the list and n = 5, then the node at index 3 in the list will jump over 5 other nodes (those at indices 4,5,6,7 and 8) and will appear following the node at index 8. You may assume that n is greater than 0 and there are at least n+1 nodes following pred.