Recitation 2

Reminder

1. I added a Q&A document for Eclipse, but Google is often your best friend.
2. You should download the github repo every week to replace the old version. Or, you can self-learn how to use Github and git.

Exercise for linked list

Note: SingularlyLinkedList is just LinkedList in our implementation.

R-3.5: refer to these lines in our course repo.

R-3.5 The removeFirst method of the SinglyLinkedList class includes a special case to reset the tail field to **null** when deleting the last node of a list (see lines 51 and 52 of Code Fragment 3.15). What are the consequences if we were to remove those two lines from the code? Explain why the class would or would not work with such a modification.

R-3.6 Give an algorithm for finding the second-to-last node in a singly linked list in which the last node is indicated by a null next reference.

R-3.9 Give an implementation of the size( ) method for the SingularlyLinkedList class, assuming that we did not maintain size as an instance variable.

R-3.12 Implement a rotate( ) method in the SinglyLinkedList class, which has semantics equal to addLast(removeFirst( )), yet without creating any new node.

C-3.17 Let A be an array of size $n \geq 2$ containing integers from 1 to $n-1$ inclusive, one of which is repeated. Describe an algorithm for finding the integer in A that is repeated.

C-3.18 Let B be an array of size $n \geq 6$ containing integers from 1 to $n-5$ inclusive, five of which are repeated. Describe an algorithm for finding the five integers in B that are repeated.

C-3.25 Describe an algorithm for concatenating two singly linked lists L and M, into a single list L' that contains all the nodes of L followed by all the nodes of M.

C-3.26 Give an algorithm for concatenating two doubly linked lists L and M, with header and trailer sentinel nodes, into a single list L'.

C-3.28 Describe in detail an algorithm for reversing a singly linked list L using only a constant amount of additional space.

Deep Equal

```
public boolean equals(Object o) {
  if (o == null) return false;
  if (getClass( ) != o.getClass( )) return false;
  SinglyLinkedList other = (SinglyLinkedList) o;        // use nonparameterized type
  if (size != other.size) return false;
  Node walkA = head;                                     // traverse the primary list
  Node walkB = other.head;                               // traverse the secondary list
  while (walkA != null) {
    if (!walkA.getElement( ).equals(walkB.getElement( ))) return false; //mismatch
    walkA = walkA.getNext( );
    walkB = walkB.getNext( );
  }
  return true;     // if we reach this, everything matched successfully
}
```

Code Fragment 3.19: Implementation of the SinglyLinkedList.equals method.

Deep Clone

```
public static int[ ][ ] deepClone(int[ ][ ] original) {
  int[ ][ ] backup = new int[original.length][ ];      // create top-level array of arrays
  for (int k=0; k < original.length; k++)
    backup[k] = original[k].clone( );                  // copy row k
  return backup;
}
```

Code Fragment 3.20: A method for creating a deep copy of a two-dimensional array of integers.