**Mason Shepherd**

**Homework 1: due January 23rd 11:59PM.**

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. **[10 points]**

Ans:

Algorithm **secondToLast**(List):

head = List.getHead() # get head to start

**if** (head = null)

**return** *“Empty list.”* # empty condition

**else** **if** (head.next = null)

**return** *“Only one element in list.”* # single element cond.

**else**

current = head # potentially secondToLast

nextAddress = head.next # potential tail

**while** (nextAddress.next != null) # determines if current is s.t.l.

current = nextAddress # move up curr

nextAddress = nextAddress.next # move up nextAddress

**return** current # secondToLast

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*′. **[10 points]**

Ans:

Algorithm **concatenateTwoDoublyLL**(L, M):

**if** L.isEmpty() **and** M.isEmpty() # both empty condition

*L*′ = null, **return** *L*′

**else** **if** L.isEmpty() # L empty condition

*L*′ = M, **return** *L*′

**else** **if** M.isEmpty() # M empty condition

*L*′ = L, **return** *L*′

**else**

*L*′ = clone(L) # create new doubly linked list *L*′ from L

x = *L*′.last() # “trailer cart” of *L*′ to be “hitched” to M

y = M.first() # “lead cart” of M which “x” is hitched to

x.setNext(y) # x -> y

y.setPrev(x) # x <- y

**return** *L*′

C-3.28 Describe in detail an algorithm for reversing a singly linked list *L* using only a constant amount of additional space. **[10 points]**

Ans:

Algorithm reverseList(L)

**if** L.isEmpty

**return** L # empty condition

**else** **if** L.getHead().next = null

**return** L # one node

**else**

curr = tail # start at back

startVar = head # start point for getPrevious()

L.setHead(curr) # back is front

i = 1

while i != L.size() # stops when new tail = old head

# --------- curr -> predecessor ---------

curr.setNext(getPrevious(curr, startVar))

curr = curr.next() # curr = predecessor

i += 1

L.setTail(curr) # curr will be the old head

**Project 1 [20 points]**

1. Compile and test the **SinglyLinkedList** class in the textbook (page 126). You are required to use your L number to test SinglyLinkedList:
2. Store your L number in the list, a node for each character.
3. Use a loop to print your L number in the linked list and report the output.
4. Turn in your complete Java code as well as the output of your program in a separate file.