Problem 1

For the singly linked list structure in MyList1.java, write a static method `append(L, M)`. If L and M are lists then the call to `append(L, M)` should return a new list W that consists of the elements of L followed by the elements of M. For example if L=[2,3,5] and M = [6,1,8,9] then the value returned would be the list [2,3,5,6,1,8,9]. The method should make a copy of each of the nodes in L, and then link the last node in the copy to the first node of M after the header. In that way, L and M are unchanged as a result of the method.

A. Write the method `append(L, M)`.

B. How does the running time of `append` depend on the length of L? How does it depend on the length of M?

Problem 2

Write a method `L.SwapN(N)` which destructively swaps the Nth and N+1st elements of a linked list. The method should use 0-based indexing, so if L = [8,1,5,6,3] and you execute `L.swap(1)`, L will now be equal to [8,5,1,6,3]. You should not create any new objects; it should just modify the existing objects.

A. Write this method for the class MyList1.

B. Write this method for the class MyList2.
Problem 3

A. Consider the example shown in the class notes of the two singly-linked lists, (class MyList1) $L=$[The quality of mercy] and $M=$[errand of mercy]. Suppose you call the method $L.DeleteAfter(L.LocateBefore("of"))$. What are the new values of $L$ and $M$?

B. Suppose that, starting from the state shown in the notes, you instead call the method $L.DeleteAfter(L.LocateBefore("mercy"))$. What are the new values of $L$ and $M$?

C. As shown in the notes, it is possible to have lists with shared structures using the data structure defined in MyList2.java. However, you cannot both have lists with shared structures and also use the “Delete” method without risking getting into an incoherent state. Explain.