**This is a “double surprise” Quiz.**

The first surprise is that there is a surprise quiz. I will tell you the second surprise at the end of the class. (My guess is that some students will find the 2nd surprise mildly annoying, while other students will like it very much.)

***THIS IS A CLOSED BOOK, CLOSED NOTES, PHONES OFF, NO INTERNET SEARCHING quiz. Anyone cheating and using any help will get zero points.***

For this “Quiz”, you are to complete the program in the Canvas assignment zip file.

The program already has code in the Form1\_Load method to create a .NET LinkedList of integers based on a starting integer array.

The program has 2 button event methods, buttonShow\_Click and buttonSort\_Click, already created.

The buttonShow\_Click method is already fully coded. If you run the program, and click it, it will write out the values in the LinkedList. The only code you should write will be in the second method.

You assignment for this quiz is to add code to the existing buttonSort\_Click method that will sort the LinkedList from low to high. When you have that working, after you click that button, then re-click the “Show List Values” button and it should write out the LinkedList values again, but this time sorted correctly.

**How to write a bubble sort**. (You must code the sort, do not use any .NET supplied sort)

First get the basic bubble sort working:

Build a nested loop, where both the inner and outer loops iterate as many times are there are items in the LinkedList.

The only thing the outer loop does is execute the inner loop the correct number of times.

The inner loop walks down the LinkedList comparing each “j” node to the “j+1” node, and if the j + 1 node is less than the j node, it swaps them.

When you have this working, and your output is nicely sorted from 1 to 50, add the optimization.

The optimization is to keep a “flag” that you set to false each time you start the outer loop. As you walk down the LinkedList in the inner loop, if ever you do a swap of two values, you set the flag to true, indicating one or more Linked List node values changed.

In the “if” part of the outer loop setup, it should check this flag, and if the flag is false, you should leave the outer loop. (Because it means you walked the loop and did no swaps, so the list is already sorted and there is no reason to keep looping.)

Test you code, and then zip it and turn it in.