**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

For each of the below questions, write a short sentence or two to express (in your own words) your answer. Keep the answers short, but use complete, correct, English sentences.

If it helps to clarify the questions, feel free to mentally prefix all the questions with the phrase "According to the video…"

1. After you’ve watched all the videos, please answer this question:  
   Of all the videos that you watched, if you could pick one video to be re-recorded by the instructor outside of class which would you choose? Why?  
   (Keep in mind the recording outside of class will omit any pauses from the instructor answering student questions, have less hemming and hawing, etc, and generally be more concise)

|  |
| --- |
| < Write your answer here > |

**VIDEO: Linked List Schema: Traversal**

Note that the document being discussed is actually listed in Lesson 05 (***Strategies for LL: Traversing A Linked List)***

1. When dividing up the work that needs to be done in order to walk through a linked list, what is the first step?

|  |
| --- |
| < Write your answer here > |

1. Based on prior exercises (possibly in prior lessons), what will the "Iteration Logic " step need to do?

|  |
| --- |
| < Etc. > |

1. What sort of work gets done in the "Teardown" step? Will you always need this step in order to traverse a linked list?

|  |
| --- |
|  |

**VIDEO: Linked Lists: InsertAt**

1. When adding something to the middle of a list, how will we determine where in the list to put it? (And what is the other way to determine where to put the new value)

|  |
| --- |
|  |

1. If the index parameter is larger than the size of the list, what will our implantation do?

|  |
| --- |
|  |

1. What are the two basic cases that our code will need to handle?

|  |
| --- |
|  |

1. What are the two general values that the **front** reference might hold?

|  |
| --- |
|  |

1. Under what circumstances to we want to add something to the front of the list?

|  |
| --- |
|  |

1. Once you've decided to add something to the front of the list, what are the steps you need to go through in order to add a new node to the front of the list?

|  |
| --- |
|  |

1. Is the variable **ln** a reference, or the actual object?

|  |
| --- |
|  |

1. Why is the question "Add something to a linked list" a good (and surprisingly common) technical interview question?

|  |
| --- |
|  |

1. How do you use this new AddAt method (in, say, main)? Provide a C# code snippet that demonstrates this.

|  |
| --- |
|  |

1. Is it normal to sort an unsorted linked list? If not, what is normally done instead?

|  |
| --- |
|  |

1. What is the C# code that provides basic pattern for walking down the list? How does it work? (For example, make sure that you could trace through this code)

|  |
| --- |
|  |

1. What is the C# code the provides both the basic list-walking logic AND stops walking at the point that will correctly leave cur pointing at the node PRIOR to desired location of the new node?  
   (Make sure that you use the version that will ensure that cur will not walk off the list)

|  |
| --- |
|  |

1. How do we know that cur is not null when we start the list-walking loop?

|  |
| --- |
|  |

1. When we end the loop, where might **cur** be pointing to?

|  |
| --- |
|  |

1. How do we add something to the list, when **cur** ends up pointing to the last item in the list?

|  |
| --- |
|  |

1. How do we add something to the list, when **cur** ends up pointing to an item somewhere in the middle of the list?

|  |
| --- |
|  |

1. What is the final, finished, complete C# code for the AddAt method?

|  |
| --- |
|  |

**VIDEO: Linked Lists: RemoveAt**

1. What is the first thing to check for? Why should the method return if this is true?

|  |
| --- |
|  |

1. What is the C# code to remove the first item in the list?

|  |
| --- |
|  |

1. If **front.next** is null, how many items are in the list?

|  |
| --- |
|  |

1. What is the C# code to walk through the list, leaving cur correctly positioned to remove the target element?

|  |
| --- |
|  |

1. What are the two possible reasons why we've exited the list-walking loop?

|  |
| --- |
|  |

1. What is the C# code to **check that we should** remove the element after cur?

|  |
| --- |
|  |

1. What is the C# code to **actually remove** the element after cur?

|  |
| --- |
|  |

**VIDEO: Linked Lists: Notes on Clone**

1. What is a very common error in the Clone method (when you're trying to make a deep copy of the linked list)?

|  |
| --- |
|  |

1. If you use the AddAt routine to add in cloned list nodes, what will the running time (in Big Oh notation) for the Clone method be?

|  |
| --- |
|  |

1. What is a better (faster) way to copy all the nodes in the list?

|  |
| --- |
|  |