**Name: Patrick O’Brien**

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: Nested Classes**

1. By default, where should you put your classes (even after watching this video)?

|  |
| --- |
| Put the classes outside of other classes. |

1. Why does it make sense to make the node class (within a linked list) into a nested class? (I.e., what rationale was provided in the video)?

|  |
| --- |
| It exists solely for the MyIntList class. Nobody else needs to make use of it or know about it.  So therefore it makes the most sense to put it inside the MyIntList class, creating a nested class. |

1. If IntListNode is marked as protected, which classes can access it? What if it's marked private?

|  |
| --- |
| Only things within the MyIntList class can access the data within the private IntListNode class.  If it is marked protected, then any subclasses connected to MyIntList class can also access it.  Nobody else outside of the class can create an IntListNode. |

1. When do you want to use a nested class?

|  |
| --- |
| Have a small helper class that you can attach methods to, but not allow anyone else to use it.  Only use when it’s a small helper class. |

1. Why is it ok to mark the data fields of the protected nested class as being public?

|  |
| --- |
| Because the node is just for the exclusive use of the list class, and it can’t be used for any other classes. There’s no reason not to mark the methods public.  You are saying that is only public within the list. You don’t have to create any getters and setters. |

1. Why does the LinkedList\_Verifier need your nested IntListNode to be protected, and not private?

|  |
| --- |
| Because it is a subclass, if it is private the subclass LinkedList\_Verifier will be unable to access it. |

1. In a nutshell, how does the LinkedList\_Verifier check to see if your linked list is correct?

|  |
| --- |
| It calls all of your add and remove routines. It has extra methods to check the array, signaling if certain things are correct or incorrect. |

**VIDEO: Linked Lists: Overview**

1. Up till now, what did you (typically) use in order to store a collection of things?

|  |
| --- |
|  |

1. What is the major downside of using an array to store items?

|  |
| --- |
|  |

1. What are the (minimum) two fields that each node in the linked list must have?

|  |
| --- |
|  |

1. What value will the last node in the list use for it's **next** field?

|  |
| --- |
|  |

1. Which node will we keep track of? (Will we keep track of all of them, or just one?)

|  |
| --- |
|  |

1. If I wanted to add something to the middle of a linked list, how do I do that (answer this intuitively/pictorially, NOT using C# code)?

|  |
| --- |
|  |

1. If I wanted to add something to the middle of an array, what would I have to do? Why is it easier to add something to the middle of a linked list?

|  |
| --- |
|  |

1. In addition to the "node" object/class that the video discusses, what other class/object does the video mention (towards the end)? What is the purpose of this second class?

|  |
| --- |
|  |

1. Are the "next" links for each node one-directional or bi-directional? If they're one-way, which way do they point?

|  |
| --- |
|  |

**VIDEO: Linked Lists: AddAtFront**

1. Why can't we use the name **LinkedList** when creating our own linked list class? What is a good name to use instead?

|  |
| --- |
|  |

1. What is the role/purpose of the MyLinkedList class?

|  |
| --- |
|  |

1. In this video, where are we going to add new items to the list?

|  |
| --- |
|  |

1. Why is it important that the **front** reference start out with the value null?  
   Is it necessary to assign null to front ourselves? Why or why not?

|  |
| --- |
|  |

1. Should the Node class be nested or separate? If nested, which class should it be nested within?

|  |
| --- |
|  |

1. What line of C# code will determine if a list is empty?

|  |
| --- |
|  |

1. When the list is empty, how do you add a new node to the list?

|  |
| --- |
|  |

1. The constructor for the Node class does not explicitly set the value of **next** – what value will next have?

|  |
| --- |
|  |

1. How can you think about the computer's memory?  
   If I say "The node's address is 70,000" what does that mean (intuitively)?

|  |
| --- |
|  |

1. C++ does allow you to actually get the memory address of objects – why does C#/Java/etc NOT allow us to get these memory addresses?

|  |
| --- |
|  |

1. When adding a new node to the front of an existing list, what three steps do we have to do?  
   (List these intuitively/in English – NOT using C# code)

|  |
| --- |
|  |

1. What is the C# source code for the three steps that you explained in the previous question?

|  |
| --- |
|  |

1. What is the difference between **nn** and **nn.** ?

|  |
| --- |
|  |

1. What will happen if you accidentally reverse the order of the second two steps?

|  |
| --- |
|  |

1. In the Node class, is **next** an actual, embedded Node object? If not, then what is it?

|  |
| --- |
|  |

**VIDEO: Linked Lists: PrintAll**

1. What does the phrase "traversing the list" mean?

|  |
| --- |
|  |

1. What is the first step in printing all the nodes in the list?  
   (List this in English, not C# code)

|  |
| --- |
|  |

1. What steps will we repeatedly do, in order to print all the nodes in the list?  
   (List this in English, not C# code)

|  |
| --- |
|  |

1. What is the C# source code to print everything in the list?

|  |
| --- |
|  |

1. Why doesn't PrintAll crash if the list is empty?

|  |
| --- |
|  |

1. Within the .Net platform library, what pattern is often used to store items in an array (instead of storing integers, like we typically do here)?

|  |
| --- |
|  |

**VIDEO: Linked Lists: RemoveFromFront**

1. When you create a new MyLinkedList class, do you automatically create a Node class at the same time?  
   When do you get new Node objects?

|  |
| --- |
|  |

1. What two things will the RemoveFromFront method do?

|  |
| --- |
|  |

1. Explain in English what should be done if the list is empty:

|  |
| --- |
|  |

1. If front is not null, what do we know about the list?

|  |
| --- |
|  |

1. List out the C# source code that will remove the first item from the list & return it's data field:

|  |
| --- |
|  |

1. If we call RemoveFromFront on a list with two items, what will the list look like after the method ends?

|  |
| --- |
|  |

1. What will happen to the nodes that no longer have any references referring to them?

|  |
| --- |
|  |

1. If we call RemoveFromFront on a list with only one item, what will the list look like after the method ends?

|  |
| --- |
|  |

1. If we call RemoveFromFront on a list with no items, what will the list look like after the method ends?

|  |
| --- |
|  |

**VIDEO: Linked Lists: Print At Location**

1. In terms of our PrintAtLocation method, what is the index of the first item in the list? What is the index of the second item?

|  |
| --- |
|  |

1. What is the correct C# source code for the loop that walks to a particular node of the linked list?  
   Why will || (the logical OR) NOT work correctly in the loop?

|  |
| --- |
|  |

1. When the method goes directly to using a Console.WriteLine on cur.data, and the given index is larger than the list, what will happen (and more importantly, why does it happen)?  
   How do we fix the program so that it doesn't crash (list C# source code)?

|  |
| --- |
|  |

**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?

|  |
| --- |
|  |