**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)

|  |
| --- |
| Audio popping on binarySearch\_example |

**VIDEO: Linear/Binary Search**

1. Let’s say that you want to determine if a particular value is present in an array.  
   Briefly and intuitively, describe how a linear search operates.

|  |
| --- |
| Linear search checks each item one at a time, each slot in the array to see if it is the right one. If the current slot is not the correct one, then it moves to the next. |

1. In the video, what does the variable **N** represent?

|  |
| --- |
| N represents the number of elements in the array. |

1. Why is this search method called “linear” search?

|  |
| --- |
| Because the amount of time required to complete it increases in a linear fashion |

1. What sort of arrays can you use a linear search on?

|  |
| --- |
| Any arrays that follow a linear method of storage (probably not a tree or a graph, though I’m not entirely sure). |

1. In addition to the properties needed for a linear search, what must also be true of an array in order to use a binary search on it?

|  |
| --- |
| Properties of being sorted into some order (ascending in the case of the video). Any order works as long as it’s well defined. |

1. How does a binary search begin?

|  |
| --- |
| Binary search starts in the middle, if it is, then it’s done. Then it’s either to the left or to the right. It clearly can’t be both. It will choose the side that has a range where the number exists. |

1. If that array element is not the target value, then what will the binary search do next?

|  |
| --- |
| Go one less than the thing that was just checked. For instance if looking for 7 and you just checked 11, you would check the next number in that range, 0-10, and you would check 10. |

1. Briefly describe how the binary search continues searching (after the initial check)

|  |
| --- |
| It will repeatedly split the array in two. It will either be to the left or to the right of the new middle. |

1. What is an important test case to check when testing a search algorithm (such as binary search)?

|  |
| --- |
| Making sure things still work when there is only one test case left. Also when the case is actually not present. |

**VIDEO: Binary Search on paper: Example Solution**Notice that in this lesson in the In-Class Exercises portion of the web page there are a number of documents linked to by **“Linear Search By Hand”** or **“Binary Search By Hand”** or “**Bubble Sort By Hand”**. This video explains how to do the “**Binary Search By Hand**” on-paper exercise.   
  
It is recommended that you try to do the exercise on your own before you watch this video, then check your work against the explanation in the video.

1. During Iteration 0, what values are tracked, and what does each value mean?

|  |
| --- |
| Target value: 17  Max Index: 15  Min Index: 0  Index ½ in between: 7  Value at that middle index: False  Next iteration change Max to: 6 |

1. At the end of Iteration 0, why does max change to 6?  
   (Also - why does it not change to 7?)

|  |
| --- |
| Does not change to 7 because 7 is the index ½ between the two ranges. It starts at 7.  Then because the value is not on the right size, it swaps to left, where the max starts at index 6 in the array, one to the left of the middle index 7. |

1. At the end of Iteration 1, what value is changed to 4, and why?

|  |
| --- |
| Min changes to 4. 4 might contain the value but it doesn’t. What happens is checked all values, checked middle on 7, intuitively need to check the values on the right hand side. Move up to min so we check 4-6. If target value had been 3, continue to move max down. |

Make sure that you check your work against the remainder of the video

1. Given how awesomely efficient binary search is, why don’t we always use binary search on every array that we want to search?

|  |
| --- |
| Because it requires the array to already be ordered. If it’s not, the binary search algorithm will not work. |

**VIDEO: Bubble Sort**

1. About a minute into the video BubbleSort is quickly summarized. According to that summary, how does BubbleSort (basically) operate?

|  |
| --- |
| BubbleSort  Run through the array a whole bunch of times, if there are two things out of order, swap them. In this manner you can run through and sort the array so that something like binary search can be used. |

1. For BubbleSort, how are numbers swapped inside the computer?  
   In contrast, how will we represent numbers being swapped when doing the paper exercise?

|  |
| --- |
| Indexes are swapped inside the computer after the two numbers are compared.  On paper you write out the entire index and swap the numbers for each new comparison. This way the changes are apparent. |

1. (Roughly) how many elements does Pass #1 compare?

|  |
| --- |
| The first pass compares all the elements in the array. |

1. In Pass #1, what happens if you find the largest element somewhere in the beginning/middle of the array?

|  |
| --- |
| The largest value is always out of order respect to the number next to it. (what?)  By the end of the first pass, the largest value will be at the end of the array. |

1. After one pass through the array, what is true about the values in the array?

|  |
| --- |
| The values are still not sorted, but the first largest number is sorted to be in order at the end of the array. |

1. In the pseudocode, what is the outer loop used for?  
   Because of this, why is it ok that you never actually use the outer loop’s counter inside the inner loop?

|  |
| --- |
| Loop N times, where N is the number of elements in the array.  It is used as a counting loop. If there are 20 elements in the array, N elements, N passes through the array.  Never use the loop’s counter inside the inner loop because the one inside handles the single pass in the array. If the counting loop was inside the inner loop, it wouldn’t loop N times. |

1. Briefly, intuitively explain one (or more) optimization(s) there were briefly mentioned in the video.

|  |
| --- |
| Make sure to optimize to check whether something is already sorted. If something is already sorted you don’t need to sort it, optimizing the time it takes to run and check the array. |

**VIDEO: Bubble Sort on paper: Example solution**

Notice that in this lesson in the In-Class Exercises portion of the web page there are a number of documents linked to by **“Linear Search By Hand”** or **“Binary Search By Hand”** or “**Bubble Sort By Hand”**.

This video explains how to do the “**BubbleSort By Hand**” on-paper exercise.   
It is recommended that you try to do the exercise on your own before you watch this video, then check your work against the explanation in the video.

1. When do you copy the current array into a “fresh” (blank) array?  
   (Remember that you do NOT copy the array over every time that you do a comparison! You’ll get a hand cramp AND run out of paper trying to finish the exercise! ☺ )

|  |
| --- |
| After you’ve swapped on set of numbers. Write out the numbers in the next array and then repeat. |

1. Between the “Starting Point” row and “Swap 0”, which two elements are swapped? Why?

|  |
| --- |
| 17 and 12, because that is where the counting loop starts. |

1. Between “Swap 0” and “Swap 1”, which two elements are swapped?

|  |
| --- |
| 21 and -3, because they are the next set to be compared and swapped in the counting loop. |

1. Between “Swap 1” and “Swap 2”, which two elements are swapped?

|  |
| --- |
| 21 and 0 |

1. Between “Swap 2” and “Swap 3”, why do we “rewind, and go back to the beginning of the array”?

|  |
| --- |
| To move the next largest number to the end of the array and continue sorting. |

1. After which rows does Pass #2 end?

|  |
| --- |
| Pass #2 ends on the second to last index. |

**VIDEO: Bubble Sort: Warning about the Null Reference test case in this week's tests**

1. The BubbleSort method is public; what does this imply about the possible values of the parameters?

|  |
| --- |
| You can’t guarantee that any set of values are going to be passed to it. |

1. Give one example of a line of C# code that would cause the program to crash if your BubbleSort method was given null as a parameter; make sure to explain why it causes the crash.

|  |
| --- |
| BubbleSort(int[]nums)  {  If (nums.Length == 0)  Return;  }  It would crash because it is trying to check the length of a null parameter (the space it is trying to check does not exist, and that is the first test.) |

1. Give an example of C# code that will check if BubbleSort’s parameter is null or not:

|  |
| --- |
| If (nums != null)  {  Return;  Then run the code that comes after. |

1. Give the example from the video that checks if the array parameter is null or else if the length of the array is zero. Also explain why that line will not crash when the parameter is null.

|  |
| --- |
| If (nums == null || nums.Length == 0)  Return;  It does not crash because nothing else in the program is run. The method returns and stops executing before the null value is used beyond the check. |

1. Is it possible to create an array that is not null yet has a length of zero?  
   If so, then give an example of C# code that does this.

|  |
| --- |
| Int[] nums = new int[0]; |

**VIDEO: Big "Oh" notation**

1. What can Big Oh notation be used to measure?

|  |
| --- |
| It’s a way to describe how much time or space an algorithm is going to be able to consume. |

1. Overall, what is the goal of using Big Oh notation? What can we use it to do?

|  |
| --- |
| Make decisions on whether it’s best to sort the array and do a binary search, or just do a linear search. Be able to make decisions before any software has been written. |

1. When in the software development lifecycle would you use Big Oh notation?

|  |
| --- |
| The planning phase of the software development lifecycle. |

1. For the 'simple example' of printing out everything in array, what will control how many times the body of the loop runs?

|  |
| --- |
| The length of the array. |

1. Is Big Oh notation good for analyzing small arrays? Why or why not?  
   What sized arrays is Big Oh notation good for analyzing

|  |
| --- |
| No, it’s not.  Big Oh is better for larger arrays.  All the things that are declared for the big oh to work could be avoid for analyzing smaller arrays, but for larger arrays, the scope of usage becomes less meaningful. |

1. Why/how do we ignore any coefficients on the leading term of the formula that exactly describes how much time the algorithm will need?

|  |
| --- |
| N \* number of times it takes to run once per the loop.  N \* 72(instructions per time per loop)  The other coefficients don’t affect the outcome as much.  O(N) N🡪 cN + c |

1. If we say that an algorithm takes O( N ) time (where N is the length of the array), what are we actually saying about the exact time that it takes to run the algorithm.

|  |
| --- |
| O (N) is cN + c, where it takes all the coefficients into account to calculate the total run time of the algorithm. |

1. Is Big Oh notation an upper or lower bound on the amount of time that the algorithm will take?

|  |
| --- |
| An upper bound on how long it might take. It could take less. |

**VIDEO: Big "Oh" notation & linear search**

1. What is one complication that arises when analyzing the run-time of linear search?

|  |
| --- |
| The target value could be at the very start of the array. So it would only take one check to find the necessary value. Therefore in this singular case, the linear search would be extremely fast as opposed to extremely lengthy as it would normally take. |

1. In analyzing the run-time of any algorithm (say, linear search), what assumption are we going to make about the values contained in the array?

|  |
| --- |
| We assume that the array contains a collection of entirely random numbers. |

1. For linear search, on average, how far do we expect to go through the array before finding the target value (or determining that the value is not present)

|  |
| --- |
| That the target value is about halfway through the array on average. |

1. When writing out the running time of linear search, why is it that we do not see the coefficient listed in the video?

|  |
| --- |
| Because O(N) hides the coefficient no matter what. |

1. When examining the Excel spreadsheet in the video, what do the values in the “Array Size” column mean?

|  |
| --- |
| The array size column represents the size of the array (number of indexes). |

1. When examining the Excel spreadsheet in the video, what do the values in the “Found” column mean?

|  |
| --- |
| How much time it takes to find the target number. |

1. When examining the Excel spreadsheet in the video, what do the values in the “Not Found” column mean?

|  |
| --- |
| The not found column is how long it takes to go through the entire array assuming you searched for a target number that was not originally present the array. |

1. **What is the running time of the linear search algorithm, in the Big Oh notation?  
   (Make sure that you (briefly) note what N is!)**

|  |
| --- |
| N is the number of elements in the array.  The running time of the linear search algorithm is directly dependent on N. If you were to double the size of the array, the running time of the linear search algorithm would also double. |

**VIDEO: Big "Oh" notation & binary search**

1. Why is it good to compare algorithms based on looking at the ‘Found’ column?

|  |
| --- |
| It predicts how long it takes to find a number using the binary search. |

1. Why is it good to compare algorithms based on looking at the ‘Not Found’ column?

|  |
| --- |
| Not Found is the absolute worst case for how it takes to find a number. Worst case is how long it could take in total. |

1. Why don’t we use binary search for every single search we want to do on every single array we want to search?

|  |
| --- |
| Because binary search requires a sorted array. It can take longer to sort the array and then search it with binary than it would with a linear search. |

1. **What is the running time of the binary search algorithm, in the Big Oh notation?  
   (Make sure that you (briefly) note what N is!)**

|  |
| --- |
| N is the number of elements in the array.  The running time of the binary search algorithm is logarithmic. It works better for larger arrays. |

**VIDEO: Big "Oh" notation & bubble sort**

1. In the video we examine the behavior of the BubbleSort algorithm. Is the implementation that we’re examining optimized in any way?  
   (Although it was not covered in this video, make sure that you remember what some possible optimizations are)

|  |
| --- |
| We are assuming that the BubbleSort algorithm is the base algorithm and not optimized in any way. |

1. When examining the Excel spreadsheet in the video, what do the values in the “Sorting Swaps” column mean?

|  |
| --- |
| The number of times the numbers are swapped before the array is sorted. |

1. If you were to graph the linear search on the same graph as BubbleSort, how would the linear search appear?

|  |
| --- |
| It would appear as an exponential graph. It would continue to increase forever for as long as the array increases. |

1. Given a 20,000 element array that you want to find a single, specific value in would it be better to do a BubbleSort followed by a Binary Search, or would it be better to skip the sorting and go straight to a linear search?   
   Explain (briefly and intuitively) your reasoning.  
   Note: There is a correct answer to this question, but because the video does not say it you will not lose points for failing to list it here. You should think about the question, reason your way to a conclusion, then concisely explain your answer (your reasoning) here.

|  |
| --- |
| It would be better to use a linear search. Because there’s so many elements in the array it would take longer to sort it and then read it then search one by one until the element is found. |

1. **What is the running time of the bubble sort algorithm, in the Big Oh notation?**(You may need to look this up online)

|  |
| --- |
| O(N2)  O(n) is the best-case running time for bubble osrt. |