**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: BST - Remove (Concepts)**

1. What is the major topic for this video?

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

1. How many cases do we have to worry about? What are they (in a nutshell)

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

1. When asked to remove the value ‘15’ from the tree in the video, what is the first step?

|  |
| --- |
|  |

1. When walking down the tree (looking for the target node that contains 15) how many pointers/references do we maintain? What are their names?

|  |
| --- |
|  |

1. Which case does removing the value ‘15’ from the tree in the video illustrate?

|  |
| --- |
|  |

1. Case 1 is similar to removing what from a linked list?

|  |
| --- |
|  |

1. Which case does removing the value ‘10’ from the tree in the video illustrate?

|  |
| --- |
|  |

1. When removing 10 from the tree, how do we remove it? Please explain this both intuitively, and by talking about the left/right references in the parent.

|  |
| --- |
|  |

1. Why is it safe to remove 10 from the tree?

|  |
| --- |
|  |

1. Which case does removing the value ‘50’ from the tree in the video illustrate?

|  |
| --- |
|  |

1. “Jamming the 25 tree under the 75 tree” will result in a tree that has what undesirable property?

|  |
| --- |
|  |

1. Will we actually remove the **node** containing the value 50? If not, how will we remove 50 from the tree?

|  |
| --- |
|  |

1. Mechanically, what path does one follow in order to find the predecessor value in the tree?

|  |
| --- |
|  |

1. In the third case (removing a value stored in a node with two children), what value should you always use in this class (unless specifically instructed otherwise)?

|  |
| --- |
|  |

**VIDEO: BST: Remove By Hand Exercise**

1. What is the basic goal of this video?

|  |
| --- |
|  |

1. Show the tree that will result from adding the values 10, 4, 6, 8, 2, 1
(Make sure that you understand why the tree looks like this, and not any other possible tree that contains all of these values)

|  |
| --- |
|  |

1. For this exercise, should you simply show the tree that results from removing the target value, or should you actually walk through the logic involved in removing the target value?

|  |
| --- |
|  |

**VIDEO: BST: Remove (Overview)**

1. What is the first case we consider, when removing a value from a BST?

|  |
| --- |
|  |

1. For the first case: In terms of searching for the target value, where do we have to stop in order to be able to remove the target value from the tree?

|  |
| --- |
|  |

1. What is the second case?

|  |
| --- |
|  |

1. In the second case, what's the (basic, intuitive) first stage in removing the target value?

|  |
| --- |
|  |

1. Once we've found the target value (in a specific node), how will we remove that value from the tree (intuitively)?

|  |
| --- |
|  |

1. Why is it safe to remove that node in the way you said in the prior question?

|  |
| --- |
|  |

1. What defines the third case?

|  |
| --- |
|  |

(The following questions all apply to the third case)

1. Instead of removing the NODE containing target value, what are we going to do instead?

|  |
| --- |
|  |

1. In order to do what we said we'd do in the prior question, which value are we going to find in the tree?

|  |
| --- |
|  |

1. What path through the tree will we follow in order to find the value described in the prior question?

|  |
| --- |
|  |

1. In the example given, do we know if the node with the value of "3" has a left child?

|  |
| --- |
|  |

1. What will we do with the value 3? What happened to the node that used to contain 5?

|  |
| --- |
|  |

1. What do we do with the node that originally contained 3?

|  |
| --- |
|  |

1. In this context, what does the word "predecessor" mean?

|  |
| --- |
|  |

1. Why is it easier (and safe) to remove the node that originally contained the value 3?

|  |
| --- |
|  |

1. How do we know that we haven't destroyed the search property on the node that originally contained the value 5?
(Specifically – how do you know that the Find method will still correctly find values greater than 5? How do you know that Find will correctly find values less than 5?

|  |
| --- |
|  |

1. In terms of the predecessor node, are there any guarantees about how many children it has?

|  |
| --- |
|  |

1. In terms of removing the precedessor node, what was significant about the situation where we wanted to remove the value 1 from the tree? How did it differ from the situation where we were removing the value 5?

|  |
| --- |
|  |

1. Instead of using the predecessor value, what other value could we use?
(Make sure that you know BOTH what the 'name' of that type of node is, and that you can describe what value that is)

|  |
| --- |
|  |

1. In this class, unless the instructor says otherwise, which node should you remove from the tree in order to correctly do the third case of BST.Remove?

|  |
| --- |
|  |

1. Assuming that the tree is balanced, what is the running time (in Big Oh notation) of BST.Remove?

|  |
| --- |
|  |

1. How do computer scientists write log2 (at least when talking about Big Oh notation)?
(Will mathematicians generally, and your math teacher specifically, accept this as valid notation?)

|  |
| --- |
|  |

1. What is the worst case structure of the tree when we're doing a remove?

|  |
| --- |
|  |

1. In the worst case, what is the running time (in Big Oh notation) of doing a BST.Remove?

|  |
| --- |
|  |

1. Does the BST.Remove actively balance the tree as it removes values?
(What are the names of two other binary tree structures that do actively balance the tree)?

|  |
| --- |
|  |

**VIDEO: QuickSort Algorithm**

1. What can you use QuickSort on?

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

1. Is Quicksort always faster than BubbleSort? If not, when will/won't it be faster?

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

1. Quickly summarize how BubbleSort works

|  |
| --- |
|  |

1. What is the running time of BubbleSort (in Big Oh notation)?

|  |
| --- |
|  |

1. How is Quicksort fundamentally different from BubbleSort?

|  |
| --- |
|  |

1. What is the pivot value? Where will we choose the pivot value from?

|  |
| --- |
|  |

1. Each time QSort gets called, how is it going to arrange the elements of the subsection of the array which QSort got called on?

|  |
| --- |
|  |

1. Is there any guarantee that the half the values will be placed to the left of the pivot, and half to the right of the pivot?

|  |
| --- |
|  |

1. Which function is responsible for choosing the pivot value? Does that function do anything else?

|  |
| --- |
|  |

1. What value does Partition return?

|  |
| --- |
|  |

1. Each time QSort gets called, where is the pivot value placed, relative to it's final, sorted location?

|  |
| --- |
|  |

1. Which value will we choose for the pivot value?

|  |
| --- |
|  |

1. How will we figure out how to swap a too-large element with a too-small element?

|  |
| --- |
|  |

1. What is the running time (in Big Oh notation) of the partition function?

|  |
| --- |
|  |

1. In the partition function, how do we determine if an element on the left is too large? How do we determine if an element on the right is too small?

|  |
| --- |
|  |

1. When does the partition function stop walking the left & right indeces?

|  |
| --- |
|  |

1. Once we've stopped walking the left & right indeces, where do we move the pivot value to?

|  |
| --- |
|  |

1. After the Partition function has rearranged the array, what function will be called next?

|  |
| --- |
|  |

**VIDEO: QuickSort Time / Space**

1. After calling Partition, where does the pivot element end up (and why)?

|  |
| --- |
|  |

1. Hopefully, how large should the left and right halves be after calling partition?

|  |
| --- |
|  |

1. Given an array of size N, how deep should the function call tree (of QSort invocations) be? (Assuming "nice" splits from the Partition function)

|  |
| --- |
|  |

1. How much time will Partition take, in total, to partition each level of the call tree?

|  |
| --- |
|  |

1. What is the running time, in Big Oh notation, of QuickSort (assuming "average" splits from the Partition function)

|  |
| --- |
|  |

1. How much space will the QuickSort algorithm require (in the "average" case)?

|  |
| --- |
|  |

1. What is the worst possible array for QuickSort to be called on (where "worst" means most time/space – consuming)

|  |
| --- |
|  |

**VIDEO: QuickSort Worst Case**

1. When the array is sorted, which element will Partition choose as the pivot element?

|  |
| --- |
|  |

1. In the worst case, how many times will QuickSort be recursively called?

|  |
| --- |
|  |

1. In Big Oh notation, what is the worst-case **running time** of QuickSort?

|  |
| --- |
|  |

1. In Big Oh notation, what is the worst-case **space required** by QuickSort?

|  |
| --- |
|  |

1. In terms of optimizing Quicksort: When the size of a slice of the array gets down to a small number (such as 5 – 10 items), what will some implementations of Quicksort do *instead* of continuing to recursively call itself? Why is this more efficient?

|  |
| --- |
|  |

1. If you were to pick the pivot element randomly, what is something that you'll have to watch out for / fix (in the implementation discussed in the video)

|  |
| --- |
|  |