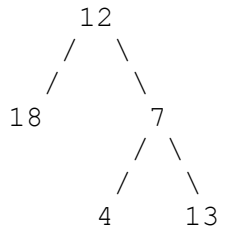
CSE143X Lecture Questions for Friday, 11/20/20

Time (e.g., 12:45)	Question	Answer
34:20	Didn't switch to the document camera on the recording for the first few minutes.	Binary tree being discussed: 
	Could we check for the base case at the end of the recursive method? I see, thank you.	You would still need to check at the beginning of the method as well, so that would lead to duplicate checks. That's why the "recursive zen" approach is to allow it to make an extra call where the check is made.
3:10	Something that I did not fully understand about recursion is why do we always have to have two methods; one public and one private. If one of the method's job is only to call the other method, we might as well eliminate this method and only call the method itself that does the job. Makes sense, thanks,	Often the key to solving a problem recursively is the selection of appropriate parameters. This is normally something you wouldn't want to force a client to understand. So it makes sense to have a public method with the kind of parameter passing you want the client to see and then you have complete control over the parameter passing for the recursion by having a private method.
	The lecture in the end got cut off as you were saying something Ok, thanks.	I'll start with that on Monday.

	How can one do the “unchoose” step when one is manipulating a String? It makes sense with an object like a Board, but what about immutable objects or primitives? But why is the unchoose necessary in that case? Okay. So it’s just a best practice to do the unchoose in case we need to use the object later? Even if the specific problem doesn’t require it? Great, thank you. Makes sense	In some of the section problems we constructed new strings. For example, as you choose between going N, NE, or E, we would append characters to a string. That works even though strings are immutable. In those cases it wasn’t because it was part of the parameter, but you might update a string, for example, and need to go back to the previous string. You shouldn’t include code you don’t need. I’ve said that the unchoose step might not be necessary. Each problem leads to its own different situations.
14:03	How many patterns could I get for the 4-queen board finally? Much fewer than I think, thx!	There are two solutions.
15	Based on the answers above, the rule for having a add-remove pair is cause b is a parameter of the recursion method? Make sense.	The backtracking solution depends on choosing positions for queens and placing them on the board. That means you have to remove them later or it just won’t work.
17:20	According to the directory you developed and the decision tree, the 8-queen recursion is implementing dfs? okie.	Recursive backtracking tends to lead to depth-first exploration. The tree is in some sense neutral because you could explore it in any order.

	When I am implementing a public-private pair, which essentially do the same thing, is it ok to do referential method comments such as the following (to avoid copying the main method comment) /** This method … */ public void doSomething() {...} /** @see Class#doSomething() */ private void doSomething(int argument) {...}	Sure, you could do that. Also we are a lot more relaxed about comments on private methods. The big requirements are on commenting the public methods that a client would see.
	Why does this happen (from output comparison tool (dict1.txt) phrase to scramble (return to quit)? 1234 Max words to include (0 for no max)? 0 [] ← why? Would we lose points for making this a special case? My current solution with no such special cases does not output anything (but works in all other cases)	There are no letters to account for in the string "1234." The question is to show all sets of words that contain that set of letters. There is exactly one such solution. The empty set. Notice that this is very different from something like the string "r" where there is no set of words that has that combination of letters. We don't comment on grading. Make what you think are good choices.
	You're probably not going to say, but would exploration being in dictionary order count as an implementation detail or a documentable fact? Ok, that makes sense. Thanks!	If it has to do with the behavior of the object, then it is something the client would want to know. If it's a detail about how it was implemented, then it is an implementation detail.
	You may want to update the documentation. It is currently written as if we are implementing LetterInventory (such as "your method should throw an IllegalArgumentException" for void set(char letter, in value))	This is an artifact of how it is used in 143. For the 143 class, they implement LetterInventory early in the quarter. But I can see how it can be confusing.
40	Can subtrees share the same leafnode or are they independent of each other?	They have to be independent.

46	When a tree only has one subtree, it could be considered as a linked list? Good example, thxx.	No, because the left and right subtrees are distinguishable. A tree with an empty left subtree and a leaf to the right is not the same as a tree with a leaf to the left and an empty subtree to the right.