Section 7 CCI summary in a more step by step format by Swapnil Asawa

Question understanding:

- Listen carefully, write it down.
- Clarify the question?
- Clarify the range, type variety of input possibilities.
- Design a few examples. A generic example, and a few specific edge cases. Large examples.

BruteForce:

- The most obvious solution.
- If no obvious solution:
  - Reduce some constraints? How can u solve a simpler problem first?
  - Base case(n=1) and build (n=2, 3,4)
  - Go through Algo list: Hash table? DP? Greedy? backtracking? Recursion? Breadth first-depth first? Binary search?
- If got a solution:
  - What's the complexity? What's the best conceivable complexity(if easy to calculate)?

Optimize:

- Any missing information?
- Bottleneck, Make it faster?
- Remove the bottleneck and design a new solution?
- Break out of for loops. Or the do-while loop. Change the nested for loops.
- Check for duplicates type of code.
- Any different approach?
- Hash table?
- Let's go through the Data structure list?
- A different example(to unclog the mind)?
- Time vs space trade-off?
- Precompute a few things?
- Best runtime vs ur time?
- How would you do it if you need to do it yourself if given with a very big example?

Planning:

- Don't code! Plan! Plan the entire algo and feel a structure of it. Write down the steps!/pseudocode.
- Think in terms of functions that this will do this and this will do that. etc.

Whiteboard.

- Top left corner.
- No slanted lines.
- Good names but not long.
- Make function names for simple things and write them down later.

Testing:

- Error checks in every function.
- Classes when appropriate.
- Conceptual test by reading each line and seeing if it actually doing what it should be doing.
- Test the code with a small example. Also the edge cases: null, the ones you designed initially.
- Double-check Weird parts like a = length-2 and see if it actually does what it should be doing.
- Solve bugs with good ways, not the fastest but shitty way.

Section 7 CCI summary in a more step by step format by Swapnil Asawa

Stress control:

You may not be evaluated on if your solution is right but the approach, how optimal, time taken, how much help needed, clean code or not, how u r compared to others,