The author is talking about pushing yourself when you practice programming and at the same time keeping it fun. For example, instead of learning how to make toy programs in main and staying in main forever push yourself to try and make separate functions and files. Personally I loose site of why I’m programming at times and I take the comfortable path by using tools I’m comfortable with. Instead of pushing myself to have a deeper understanding of a tool that would be more beneficial in the long run. I think the way it applies to becoming a great computer scientist. is that they are never satisfied with where they are at and always pushing to learn something new or do something better. Of course I’m not a great computer scientist so I’m not sure if what I just said is at all accurate or not. But regardless I think deliberative practice is essential to becoming a great programmer.

The author was illustrating what level you are at your craft before and after you become a ‘master’ at what you do. He is explaining how you can’t truly be at your best until you have put in the time. Which makes sense and I think that this can be applied to anything you do I know none of the first 10,000 lines of code I write will be my best. None of my first 10,000 possessions in basketball were my best. Some outcomes were better than others just like they will be in CS. But overall I am better. Kind of like in a video game where even if you get a perfect score on easy its not as good as a average score on hard. The outcome was better but no action on easy will be as worthwhile as it would be on hard.

The number one recipe that I can most relate to is the one about working with other programmers. Because in some small school projects or homework assignments I have been the leader in the group and I had to push myself and kind of learn by doing. And with my internship this summer, the other intern is a better programmer than I am and is the leader, at the internship I am not so much pushing myself more so learning forms the other intern and asking a ton of questions trying to learn the do’s and don’ts and how he comes to the solutions that he does so I can use those approaches along with my own in the future.

The other recipe that I feel is applicable to me is keeping it fun. Sometimes I get tired of coding and CS is not fun, when that happens I will start messing around with unity or watching YouTube videos of products that were made through CS and that gives me energy and reminds me why I found CS so fun in the first place.