In the conclusion, I want a critical reflection on the content of the course. Step back from the technical details. How does the course fit into the wider world of programming languages and software engineering?

To make a poor analogy, this course is like a construction worker learning physics. Technically, a builder can put a building together and follow the instructions on the blueprints without knowing why the engineer did any of it. The problem with the builder working like that is he has no way of utilizing his building skills without someone telling him what to build. This is analogous to the life of a software engineer because yes, technically any 12-year-old can create a basic video game in python by following a YouTube tutorial, but there is so much more to programming than following instructions.

This course allows us to go from basically utilizing the tools provided to us to being able to create our own tools to better tackle the next problem we face. In my life this has been made very clear through talking to my dad about the problems he faces on the day to day. My dad is a UCLA graduate with a degree in computer science and he said that Compiler Construction is the most useful class he’s taken. In his work, he created a parser to automatically extract information from and categorize his employer’s mess of custom stock trade orders. As I took this class, we were able to talk more and more about how his parser works and why he does what he does. I am able to understand the problems he faces and how I might be able to tackle the same solutions.

This class was also incredibly effective at teaching me how to use the basic tools to create more complex ones. This class hammers recursion into our brains. Recursion has always been an incredibly challenging topic for me. Challenging to create. Challenging to debug. Challenging to read. This class has taught me how to approach recursion and the kinds of problems that can best be overcome using recursion.

This class is a fantastic class to follow up from taking Computer Architecture my previous semester. Over the years I’ve been learning about how to create software and how to build solutions, but I never understood how a computer actually worked. I finally took Computer Architecture and then I was still left with questions. Thankfully less questions, but still questions. The biggest one was “so I can build software solutions and I know how we turned a chunk of silicon into a fancy math machine… now how does my code get to the assembly for the computer?” This class has now filled in the whole stack. From keyboard stroke to transistor, I have an (admittedly a general and high level) understanding of every step in between.

I think this class has allowed me to better understand the strange quirks and mishaps that occur when programming. I am far more knowledgeable on why some syntax just doesn’t work. I am far more capable of predicting what certain code will do before I run it, because I now (at least mostly) understand how it is actually being computed. As a software engineer, I am now able to better understand the tools at my disposal, and I am able to better apply myself to any problem. I’ve graduated from the builder to the engineer, the programmer to the software engineer.