Carter Roberts 12/10/2025
Loyola University New Orleans COSC-A361-001: Languages & Paradigms
Final Project: “Variable Binding & Scope Analysis”

Reflection & Conclusions

I spent a great deal of Thanksgiving Break deliberating about what type of final project I would do for this course. I did not have the freedom of schedule to do something enormous like an entire python game or a whole functioning new-language BNF tokenizer/parser, but I really wanted to do *something* that showed knowledge and kept my brain freshened on the concepts covered in the class. I took so long deliberating that the week before finals came, and I was without an idea of my own! It was a very frustrating failure of mine, I had thought at the time. I admit that a weakness of my previous programming knowledge and projects was understanding scope, so I was doubly worried about taking up this final project; I worried it would not seem sufficiently advanced, and I worried I would also fail the most basic task of actually producing the project deliverables. I worried I would only have confirmed all my old fears that I’m really not made for computer science, that I’m a little idiot among titans of peers who’ve done this forever and that I’ve hopelessly wasted good money on my time at college. Not that I’m anything too sharp or even genuinely talented at all (I’m not), but my fears evaporated when actually performing the work for this project.

Having to create succinct, strong impressions of technical concepts involving scope and binding in multiple languages impressed a great deal of technical confidence and practice into me that I sorely needed and definitely appreciate. I had to learn a new language, as well, in order to display real code snippets in a dynamically-scoped language. Emacs Lisp was sort of irritating at first because of how sheerly different the basic syntax is and because of the hotkeys in the interface of Emacs itself, but I found the language’s little knobs and bells genuinely fascinating and I think I’ll continue to read up on and practice it after the end of the semester. I’m glad that the primary focus of my project for a course entitled “Languages & Paradigms” was to actually learn the

fundamentals of a new programming language and to consider the differences between multiple programming paradigms; that of dynamic-vs-static scope and binding. I feel as though I have emerged at the opposite end of a dark tunnel of nerves and insecurities with a certain degree of enlightenment and eagerness. All of the code snippets were exhilarating exercises in trying to create something both conceptually captivating and also easy to understand from another person's perspective. The analysis was intimidating at first, but by the time I had finished the snippets and studied sufficiently, it was no sweat at all. My scope diagrams were perhaps flimsily-drawn, but I feel as though the actual explanation contained within them was proper, consistent, and communicative without too much text and jargon. The comparison table took some consideration on what to make each row discuss, and I *do* feel some lurking concerns that I was a bit redundant, but I'm able to hold onto hope as of yet and I can't think of much else to discuss on it.

Despite not actually creating an entire program, I feel that I've become a much more rounded programmer for the knowledge I've refreshed and sharpened through this project than I would have if I completed a python game/visual or some kind of C++ storage program. Not to say I'm not interested in doing those down the line, I very much am, but I was really grateful here to participate in something that was more of a brain-tease that reckoned with one of my weaknesses in my work, rather than an enormous one-week undertaking that ends up too labyrinthine to explain without losing my breath, yet imparts no fundamental and technically low-level insights upon me. One of my greatest interests in computer science is to simply unravel what to me remains a somewhat mystified device on the most basic levels; sure, I took Computer Organization, Operating Systems and Data Structures and I know about stacks, binary logic, kernel & system calls, etc.. But I feel as though all of it has become somewhat obstructed and arcane in the chaotic midst of all the higher-level programming, and you lose track of the time to investigate all the way down to the hardware operations. I was very, very grateful that this assignment gave me a chance to get more knowledge on a more primitive level and look into *how*

languages do what they do. I even patched up a weakness of my own programming skills doing so, since scope wasn't my strongest point and I used to constantly find my errors were related to misunderstanding it!