**PARSER PROJECT RESOURCES**

@hhp3. (2021, February 7). *Recursive descent parsing*. YouTube. https://youtu.be/SToUyjAsaFk?si=Q4MKkMCro3F0uuSX

Notes: Watched to understand the basic structure of a recursive descent parser.

Barland, I. (2018, November 17). *T0-parse-racket*. YouTube. https://youtu.be/IEzUFNFU880?si=lF8PcOEX1jle-DKg

Notes: Used an overall idea of recursive descent parsing and got ideas of how to code it in Racket. The actual code used many setting functions, so the code given couldn't really be used.

Felleisen, M., Horn, D. V., & Barski, C. (2013). *Realm of racket: Learn to program, one game at a Time!* No Starch Press.

Notes: Read chapters 1 - 4. Used to understand the basics of Racket and practice writing simple Racket programs.

GfG. (2023, June 9). *Recursive descent parser*. GeeksforGeeks. https://www.geeksforgeeks.org/recursive-descent-parser/

Notes: This provided a very simple grammar and a top-down parser program written in C++. Though not a functional programming language, the structure of the functions in this program made the most sense to me, so I used that structure and tried to implement it into my program.

Hare, B. (2024, February). A simple sample grammar, with parser.

Notes: My tokenizer and the basic structure of my nonterminal functions were based on this example. I changed the structure a fair amount, coding what made the most sense to me, however, my program ended up not working out as a whole. This example code also gave me a good idea of how to use "either" types.

Membreno, K. (2024, February). Parser Code from Classmate (Kevin Membreno).

Notes: I asked Kevin for help because I had no idea how to even begin programming this project. He walked me through his code and sent to it me so I could use it as a reference. I didn't end up using any of his code, because I wanted to use "either" types and write my functions in a way that made the most sense to me, which would result in a very different structure from his. It did give me a better idea of how to organize the program and how to tokenize and use regular expressions.

Racket. (n.d.). https://racket-lang.org/

Notes: Used to understand Racket syntax and explore functions.

Scott, M. L. (2016). *Programming language pragmatics*. Morgan Kaufmann.

Notes: Used Sections 2.3 and 11.3 to get a better understanding of recursive top-down parsing and how to code in a functional programming language style.