Johnathan Picek

Dr. Eric Chou

CPSC 46000-003

Programming Languages

May 1, 2022

Term Project

Technical Report

The Term Project for this class was a considerable challenge and I feel like a more competent programmer, designer, and developer after completing it. I set out originally intending to implement both lex and yacc in modified forms. As intended, I implemented a slightly modified version of the ply.lex template calclex.py as STPLexer.py to handle tokenizing and lexing as expected. Instead of using yacc, I designed a separate parser-like component to handle the grammar rules of this language. This was one of the challenged I faced and worked around. I found Lex and yacc documentation to be somewhat lacking, and I was unable to fully comprehend the design of grammar rules in yacc to be able to set the grammar rules as I wanted despite the slides, lectures, and outside resources. Instead, in STPParser.py, I used an expression stack and manually designed the logic to ensure proper functionality of each of the limited number of accepted commands and errors related to rejected entries.

Aside from the Lexer and Parser, the other two Python files in my project are the Thermostat class in STP.py, and the driver program / command line interface STP-CLI.py. which is how the project should be interacted with presently. Surprisingly, both the lengthiest (in line numbers) and most time-consuming code exists within the STP Class design itself. The Lexer, Parser, and CLI were successes with rather few hiccups along the way. The design of the Thermostat class required all the functionality written in the design document, which ended up being a little ambitious. My design was modelled after a hybrid of our current semi-smart Thermostat and some research into the Google Nest that I have recently done. In it, there are 4 time periods with individually programmable schedule settings named after the human behaviors commonly related to those periods of the day: SLEEP, WAKE, AWAY, and HOME. Presenting the schedule and boundary time information clearly in a manner that made sense in the CLI proved to be one of the more difficult aspects of it. By default, SLEEP starts before 0 and it would be irregular and awkward to display SLEEP as two separate groupings in the schedule. So I had to design logic to start the schedule output from the HOME -> SLEEP boundary (default 23) then count to the SLEEP -> WAKE boundary (default 7) which provided was a unique problem and I was eventually successful.

Within the related PowerPoint Presentation, I review and summarize these successes and challenges. I also include screenshots showing both proper and improper command entries, resulting in both the expected execution of the commands and the expected error messages, respectively. Technically, 6 commands were implemented based on 5 command keywords. Another 6 keywords were used in conjunction with the most important SET command to be used as target time periods (NOW, ALL, SLEEP, WAKE, AWAY, HOME). While my solution is functional, I am also confident that there are better approaches to both the design of a Smart Thermostat, and to some of the programming approaches I used. Regardless, I am grateful for the experience I had in building this project from start to finish and all the investigation and research along the way, and I am proud of my result.

Further work to improve this project could incorporate a step back and re-integration of yacc and grammar rules within it. A proper menu system and interactable User Interface (like what a real product would use) could be designed, as those concepts were only briefly touched on in the Hardware Controls Map, instead of the Command Line Interface used by this Prototype.