Jay Offerdahl - 2760730
Lab Time: Friday, 4:00pm - 5:50pm

Lab 4 Report

1. Speaking generally, C-like languages treat whitespace identically, regardless of how it's created. One tab or four spaces, it matters not. This is not the case for many other languages (Python, Haskell, YAML, etc.), where the construction, and placement, of whitespace is important. What complications does this introduce to our lexer if we need to account for such concerns? As a hint: consider writing a lexer for a Makefile, where only tabs and not spaces are allowed. How do the rules you have written change?

The complications introduced to our lexer obviously surround the idea of scope. If I was parsing a python program, I would need to keep track of the current level of my scope, whether I was in the third of five nested if statements, or simply inside a for loop. I would also need to ensure that scoping of entire functions was maintained. In the program I have written, a new line character (\n) simply reduces to nothing because of the fact that whitespace doesn't matter in C. However, in one of these whitespace dependent languages, a newline is a very important occurrence because it may be the indicator for a new function body or different level of scope. This approach may be hard to maintain, however, as the level of nested scope is unlimited. We would have to write our rules to be dynamic in nature so they know what scopes are still open simply by reading in the amount of whitespace in front of the next statement.

In the case of a Makefile, the rules would have to change so they recognize the number of tabs in front of each rule's body. For example, I could create a state for PRESTATE and INSTATE, where PRESTATE would be activated whenever a new rule is defined (e.g. "all: lexer") and INSTATE would be activated as soon as one tab character was read. If any other input was detected in PRESTATE, an error would be thrown.