Lab 7 Report

EECS 665, Nicholas Jurden 2415098

The main objective of this lab is to observe adding and removing variables from a symbol table, thus illustrating how variables come into and leave the scope of a program during execution. The lab parses a program, and the portion that we've implemented keeps track of the nesting of blocks within the program and keeps track of what variables are declared, their types, and their scope presence. When the block ends, we dont care about these variables anymore, and so we implement a way to record their values and print them (to ensure we've added them correctly) and then remove them from our symbol table. This is similar to how a compiler would behave. It uses this symbol table to check and see if a variable declaration is valid in a certain scope, and keeps track of its instantiations and changes in value throughout that variable's lifespan.

I had a hard time conceptualizing this lab at first. Once I got some more feedback on what exactly was happening, it helped to tie all of that back into my conceptual knowledge of symbol tables. Because of some discrepancy about what exactly constituted a block, my debugging mostly occurred in `sem_sym.c`, where I experienced issues with blevels being too high in some cases. After moving around my calls to `enterblock()` and `leaveblock()`, I was able to find the right solution. Biggest thing during this process was recognizing that the fhead was recognizing if statements as well as function heads, so

calling `enterblock()` for `fhead` resulted in incorrect level tracking. I moved the `enterblock()` call to `fname` and that corrected the issue, because `fname` is unique to functions.