CS 5500 – The Structure of a Compiler
HW #3

- This assignment is **due by 11:59 p.m. on Wednesday, Sept. 25, 2019**.
- This assignment will be worth **8%** of your course grade.
- A grading rubric is posted on Canvas so that you can see approximately how many points each functional requirement is worth.
- You can work on this assignment **with at most one other person** enrolled in this course.
- You are strongly encouraged to **take a look at all of the sample input and output files** posted on the Canvas website **before** you actually submit your assignment for grading.

Basic Instructions

For this assignment you are to modify your MIPL syntax analyzer to make it also do **symbol table management**, which is a prerequisite to doing full blown **semantic analysis** (which you'll do in HW #4). Again, you can use any programming language as long as your program compiles/interprets and executes on the cslinux machines. You are **NOT** allowed to use ***flex***, ***bison,*** or ***antlr***; you will receive **NO CREDIT** for this assignment if you do so – and we will be checking for that!

In this assignment **no attempt should be made to recover from errors**. Every error message should **include the line number** where the error occurred. Listed below are **two new errors** that your program will need to be able to detect for MIPL programs:

Undefined identifier
Multiply defined identifier

Note that you are **not** supposed to check for any other type of semantic error at this time!

The program should **NOT output the tokens and lexemes** that it encounters in the input file, **or the productions being processed** in the derivation (although that output may be useful for debugging, so you should probably leave it in your program and be able to turn it on/off). In order to check whether your symbol table management is working correctly, your program should **output the message ">>> Entering new scope" whenever it begins a scope, "<<< Exiting scope" when it ends a scope, and "+++ Adding … to symbol table with type …" whenever it makes an entry in the symbol table** (where the first … is the **name** of the identifier it is adding to the symbol table and the second … is the **type** of the identifier). In later parts of this compiler project you will want to have the ability to suppress the output of symbol table management messages,

so create an internal Boolean variable that you can set to turn on/off this functionality; do NOT use a command line argument for this purpose as our grading script will not be looking for that!

Sample Input and Output

Sample input and output files are posted on Canvas. Note that these files were created on a Windows PC, so you might need to run *dos2unix* on them before using them on a Linux/Unix machine. With the exception of whitespace and capitalization, the output produced by your program **MUST** be **IDENTICAL** to what our "grader" program produces! Use exactly the messages, etc. that are used in the sample output files. Otherwise, we reserve the right to **NOT** grade your program! As with the previous homework assignments, you might find it helpful to use the ***diff*** command to compare your output with the sample output posted on Canvas.

What To Submit For Grading

You should submit via Canvas a single **zip** file containing **only source code files and a *make* file** for your homework submission (i.e., no data files!). Name your zip file using your last name followed by your first initial (e.g., Homer Simpson would name his **simpsonh.zip**).

If you work with someone, instead name your zip file as the combination of each of your last names. For example, if Daffy Duck and Bugs Bunny worked together, their submission would be named duckbunny or bunnyduck. Also, if you work with someone, **only submit under ONE person's username**, **NOT both! We don't want to grade your program twice!**

***WARNING*: If you fail to follow all of these instructions, the automated grading script may reject your submission, in which case it will NOT be graded!!!** ☹