CS 5500 – The Structure of a Compiler
HW #7

- This assignment is **due by 11:59 p.m. on Monday, Nov. 18, 2019**.
- This assignment will be worth **15%** 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

You are to augment the code for your MIPL semantic analyzer (HW #4) to have it also do **code generation** (i.e., in this assignment you will finish building your MIPL compiler). If no lexical, syntactic, or semantic errors are encountered, your compiler should **output OAL code** for the input source program. That output can then be executed using the OAL interpreter (the source code for which is also posted on the Canvas website with the Lecture Notes).

Review the lecture handouts on OAL code generation for information about the OAL instruction set, format of the OAL code, etc.

Your program should **NOT** **output the tokens and lexemes** that it encounters in the input file, **or the productions being processed** in the derivation, **or the symbol table management output** (although all of that output may be useful for debugging, so you should probably leave it in your program and be able to turn it on/off). Your program **must** work on one of the campus Linux machines. You should provide us with a make file that will build (i.e., compile/interpret) your MIPL compiler source code (like it did for HW #4) and produce an OAL file named **oal_source.txt**. Our grader script will then do the following:

```
flex oal.l
bison oal.y
g++ oal.tab.c -o oal
oal oal_source.txt
```

Our grader script will supply any input that the executable program is expecting (e.g., if the source program has a *read* statement and expects keyboard input).

Sample Input and Output

As mentioned above, you should **suppress the output of token information, productions being applied in the output, symbol table management**

messages, and the "Completed parsing" message for this assignment. Unless errors are detected, **all we should see in your compiler output is OAL code**. As in previous homework assignments, you are advised to use some internal Boolean variable (**NOT** a command line argument!) to "turn off" any other output; for debugging purposes you will want to be able to "turn on" this output.

Sample input and output files (both OAL code output and "OAL assembled" result output) are posted on Canvas. Note that these files were created on a Windows PC, so you may need to run *dos2unix* on them before using them on a Linux/Unix machine. With the exception of whitespace and capitalization, the output generated by **running *oal*** on the OAL code that your compiler produces **MUST** be **IDENTICAL** to what our "grader" program produces! Note: Your OAL source code doesn't have to be identical to ours, only its output.

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!). Do **NOT** include the **oal.l** and **oal.y** files in your submission! 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!**