CS 5500 – The Structure of a Compiler
HW #4

- This assignment is **due by 11:59 p.m. on Friday, Oct. 4, 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

For this assignment you are to modify your code from HW #3 to make it do complete **semantic analysis**. 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!

As in HW #3, **no attempt should be made to recover from errors**. Every error message should **include the line number** where your compiler detected the error. Listed below are the **new errors** that you must be able to detect:

- **Procedure/variable mismatch**
- **Start index must be less than or equal to end index of array**
- **Expression must be of type boolean**
- **Expression must be of type integer**
- **Expressions must both be int, or both char, or both boolean**
- **Expression must be of same type as variable**
- **Input variable must be of type integer or char**
- **Output expression must be of type integer or char**
- **Indexed variable must be of array type**
- **Array variable must be indexed**
- **Index expression must be of type integer**

Sample input and output files are posted on the Canvas website. They give examples of every type of semantic error that you need to be able to detect.

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).

Regarding legal expression types for the various operators in MIPL, for the **binary arithmetic operators** (i.e., **+**, **-**, *****, ***div***), both operand expressions must be **integer** (and the resulting type of the expression would be **integer**). For the **binary logical operators** (i.e., ***and***, ***or***), both operand expressions must be **boolean** (and the resulting type of the expression would be **boolean**). For the **binary relational operators** (i.e., **=**, **>**, **<**, **<>**, **>=**, **<=**), both operand expressions must be of the **same type**, and must be **either char or integer or boolean** (and the resulting type of the expression would be **boolean**). For the **unary logical operator *not***, the operand expression must be **boolean** (and the resulting type of the expression would be **boolean**). For the **unary arithmetic operators +** and **-**, the operand expression must be **integer** (and the resulting type of the expression would be **integer**).

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!!!** ☹