CS 5500 – The Structure of a Compiler
HW #5

- This assignment is **due by 11:59 p.m. on Friday, Oct. 18, 2019**.
- This assignment will be worth **7%** 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.

Project Requirements

Posted on the Canvas website are *flex* and *bison* files that perform lexical and syntax analysis, as well as symbol table management for the following grammar:

P → **var** V **{** C **}**

V → **id** N **;** V | ε

N → **[intconst]** N | ε

C → S **;** C | ε

S → A | F | W

A → **id =** E | L **=** E

F → **if (** B **) then** S **else** S

W → **while (** B **)** S

E → E **+ intconst** | **id** | L | **intconst**

L → id **[** E **]** | L **[** E **]**

B → E R E | **true** | **false**

R → **> | < | != | >= | <=** | **==**

The only scalar data type in this language is **integer**. Each identifier name consists of a single alphabetic letter. Arrays can have n dimensions where n ≥ 1, and use zero-based indexing.

Shown below is a sample program derivable from this grammar:

```
var
x; y; z;
A[4];
B[2][3];
{
x = 2;
```

```
B[1][2] = x;
A[x] = 5;
while (x < 100) x = x + 1;
if (A[2] <= A[B[x][y]]) then x = y + 1 else z = 10;
}
```

You are to modify the given *flex* and *bison* files to make a **C++** program that will **generate intermediate (3-address) code** for any input program that can be derived from the grammar. Your intermediate code generation must be consistent with the in-class discussions on this topic.

For arrays, you must generate 3-address code to compute the corresponding row-major ordered 1D index (even if the original 3-address code is just a 1D array reference). Use the following assumptions: (1) arrays are **zero-based**, (2) every array declaration uses a **base address of 0**, and (3) an integer has a width of **4 bytes**.

The intermediate code that you generate from your program will be optimized in the next homework assignment. You do **NOT** have to build **a directed acyclic graph (DAG) of instructions** in this assignment, but you will probably need to **store the generated statements as a list of triples or quadruples** as the parsing is taking place.

For the code that you generate, start numbering your **temporary variables at one** (e.g., t1, t2, …) and your **labels at one** (e.g., L1, L2, …). When printing the 3-address statements (i.e., the output for your program), **print each statement on a separate line, including labels**. Also, for 3-address *if* statements, you are to use the syntax ***if x == false …*** instead of ***ifFalse x*** …, and ***if x == true …*** instead of ***ifTrue x*** …

Note that you are **not responsible for doing any error checking** (or error recovery) for this program; that is, you may assume that the **input files do not contain any lexical, syntax, or semantic errors**.

If your *flex* file was named **hw5.l** and your *bison* file was named **hw5.y**, we should be able to compile and execute your program on one of the campus Linux machines using the following commands (where *inputFileName* is the name of some input file):

flex hw5.l
bison hw5.y
g++ hw5.tab.c -o hw5
hw5 < inputFileName

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 may have 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! Otherwise, the grading script will say your

program is incorrect and we will have to grade your program manually to determine partial credit, which is a painstakingly long process that makes the grader very grumpy! ☹

What To Submit For Grading

You should submit via Canvas a single **zip** file containing your ***flex*** and ***bison*** files **as well as any .h files necessary for your solution**. Note that a *make* file will **NOT** be accepted for this assignment (since that is not what the automated grading script is expecting); your *flex* and *bison* files must ***#include*** your .h files as necessary.

Name your *flex* and *bison* files using your last name followed by your first initial (e.g., Homer Simpson would name his files **simpsonh.l** and **simpsonh.y**). Also name your **zip** file accordingly (e.g., **simpsonh.zip**). If you work with someone, instead name your files 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!**