COSC 445 Compiler Project #2

Distributed: 21 March 2016 **Due:** 8 April 2016 (Friday)

Precis: Manage a symbol table.

Introduction: Consider a subset of the C language:

(1) Types restricted to
`int`, `char`,
array of `int`, array of `char`
pointer to `int`, pointer to `char`.

(2) Functions have 0 or more input parameters and return any of the allowed types or void.

(3) A single file can have multiple functions. A single function can have multiple nested blocks.

(4) Variable names and function names follow the rules of C syntax.

(5) Type declarations follow the rules of C syntax.

(6) Keywords in C cannot used for variable or function names.
E.g., the statement

```
int for = 3;
```

will not produce a symbol table entry for variable `for` because `for` is a keyword in the C language.

(7) Block structure, delimited by '{' and '}' follow the rules of the C language.

(8) Maximum nesting level is 32.

Input: one source that would successfully pass lint (comprising one program).
Individual source file does **not** have to compile, but it does have to pass lint.

Note: this successful pass of lint is a *data hygiene* requirement. Your program may assume that any input to it will pass lint. Your program should not also execute lint.

Do NOT use any variables or functions that are defined externally, e.g., do not use stdio.h (printf, …)

Output: symbol table for the program. The output will include the entire symbol table and must be nicely formatted and readable.

Project:

(1) Can be written in <u>any language</u>, including C, C++, Java, Python (also including perl).

(2) Project should execute from command line or via a GUI. Running the project within the development will have points subtracted. (Note, this means that if you use a GUI, you should instantiate the GUI from the command line)

Symbol Table information

(1) For primitive variables:
- name
- type
- name and lexical level of declaring procedure (level 0 can be "global")
- the line counter value where the variable was declared

(2) For function names:
- name
- each parameter's name and type
- number of parameters
- return value type
- the line counter value where the variable was declared

(3) For array variables:
- name
- type
- name and lexical level of declaring procedure
- number of dimensions
- upper bound of each dimension
- the line counter value where the variable was declared

(4) For pointers:
- name
- type (of the dereferenced pointer)
- name and lexical level of declaring procedure
- the line counter value where the variable was declared

Tests:
(1) Use the C program p 257
(2) I will produce a test program ASAP
(3) You are responsible for exhaustively testing your code.

Turn in

Hard copy of source code
Hard copy of test/execution results
I may ask for code walk thru and/or demo with on the fly modifications

Grade based on

Meeting specs
Code quality (elegance, readability)