Project 2: C-- Compiler - Scanner & Symbol Table

DUE DATE: April 8, 2019

In this assignment, you are required to write a C-- using the Lex/Flex scanner generator tool. You are also required to use basic symbol table manipulation routines (lookup(), insertID() and printSymTab()) provided to handle all identifiers. Tokens to be scanned by your scanner are all valid symbols that you would expect to see in a C-- program. There are five classes of tokens to be handled: 1) operators, 2) separators, 3) identifiers, 4) reserved words and 5) constants (See the C-- specification for the five classes of tokens). Comments are considered as white spaces, and should be stripped off. However, in this assignment, you are required to define comments as a token, just to practice RE writing skills. The rules for token definitions can be derived from the description of the C-- language. Although your scanner should be designed to work with your parser (which will be implemented in HW3), in this assignment, your scanner only needs to handle the following jobs:

a) Print each comment captured by your token RE (regular expression).
b) Print the frequency of each identifier

Note:
1. Install flex & bison first
2. C-- Spec : Please refer to file "2018C—"

An example program is shown below along with the results that your scanner is expected to print out.

```
=============program start===============
int main()
{
/* 1: this is * just a / sample */
int n;
int abc, def, main_1;
int a1;
float b1, z_123_x_45;
write("Enter a number");

/* 2: this is a 3-line
        comment
*/
n = read();
abc = n + 1;
def = abc * abc;
write(def); /* 3: this is /* the end ***/
}
=============program end===============
```

The output from your scanner looks like the following:

```
/* 1: this is * just a / sample */
/* 2: this is a 3-line
        comment
*/
/* 3: this is /* the end ***/
```

Frequency of identifiers:

a1	1
abc	4
b1	1
def	3
main	1
main_1	1
n	3
read	1
write	2
z_123_x_45	1

Note:

a) It is required that the outputs be sorted. You could simply use the strcmp library function to do the sorting. In this assignment, the entire " (double quote) delimited string is treated as a single token i.e. a string constant. In the case of some unknown tokens, your scanner should issue an error message, and print out the unrecognized token, then exit.

b) You may assume the identifier names will not exceed 64 characters. However, the number of distinct identifiers should not be limited.

c) In the hw2 directory you may find the following files:

1) src/lexer.l	the lex template code that you may start with
2) src/symboltable.c	contains symbol table manipulation routines
3) src/header.h	contains a sample symbol table structure
4) src/Makefile	
5) test/	a test directory containing some sample tests

Submission requirements:

1) DO NOT change the executable name (scanner).
2) Compress all your files as **studentID_hw2.zip**. Then upload to Ceiba.
3) We grade the assignments on Ubuntu 16.04. Before summiting your assignment, you should make sure your version works fine on the environment.
4) You are free to modify Makefile. But make sure your make command works correctly.