COP4020p
Project 1 (proj1)
cscan - lexical analysis for C compiler

- *cscan* is a lexical analyzer for a C compiler. It identifies the C tokens from its standard input and writes them to its standard output, one per line. Afterwards it prints the number of occurrences of each type of token (`number, ident, char, string`, or the lexeme for the remaining types of tokens) in descending order. Ties should first be broken by **the string length of the token name** and then by **lexical order (phone book order of the token name)**. It writes invalid characters (with a diagnostic) to its error output instead of its standard output. Tokens are defined below. Terminals are enclosed between dittos (double quotes). Dittos are escaped inside terminal strings by preceding them with a backslash.

```
null  =
quote = "’"
ditto = """
back  = "\"
octal = "0" | "1" | ... | "7"
octch = back octal (octal | null) (octal | null)
digit = "0" | "1" | ... | "9"
alpha = "a" | ... | "z" | "A" | ... | "Z" | "_"
schar = any ascii character except quote, ditto,
        carriage return and back
char  = ((back | null) (schar | ditto)) | back back
      | back quote | octch
str   = ((back | null) (schar | quote)) | back back
      | back ditto | octch
token = digit+                         # number
      | alpha (alpha | digit)*         # ident
      | quote char quote               # char
      | ditto str+ ditto               # string
      | "(" | ")" | "," | "." | ":" | ";" | "?" | "[" | "]"
      | "{" | "}" | "̃" | "&&" | "||" | "++" | "--" | "->"
      | ("|" | "^" | "&" | "+" | "-" | "%" | "*" | "/" |
         "=" | "!" | ">" | ">>" | "<" | "<<") ("=" | null)
```

Blanks, tabs, newlines, and comments (enclosed between "/*" and "*/") are ignored between tokens, and escaped newlines are ignored in strings. Note that defines, includes, and hex, long, and real constants are not handled. You should comment your program so that others (e.g. the grader) can understand it.

You should also have comments at the top of the file indicating **your name**, this course, the assignment, and **the command used to compile your program.** For example:

```
/**********************************************
 *                                            *
 * Name: Derek Yohn                           *
 * Class: COP4020                             *
 * Assignment: Proj 1 (Implementing a C scanner) *
 * Compile: "gcc -g -o cscan.exe cscan.c"     *
 **********************************************/
```

- The file `~uh/cop4020/proj1/sample.c` on *linprog.cs.fsu.edu* contains an example C code as shown below.

```
/* Example Program */

main()

{

  printf("This is a test.\n");

}
```

- The following output will appear when the above program is input to `~uh/cop4020/proj1/cscan.exe` on *linprog.cs.fsu.edu* as follows.

```
%~uh/cop4020/proj1/cscan.exe < ~uh/cop4020/proj1/sample.c

main

(

)

{
printf

(

"This is a test.\n"

)

;

}
                       count
         token
--------------------     -----
              ident      2
                  (      2
                  )      2
             string      1
                  ;      1
                  {      1
                  }      1
```

- Note that I used the following C printf format for the token count report:

  ```
  printf("         token          count\n");
  printf("----------------------  -----\n");
  printf("%21s   %5d\n", lexeme, tokencount);
  ```

  Your output should match my format exactly so grader (mentor) and I can check your cscan program for correctness using the *diff* UNIX command.

- You ***MUST*** implement this assignment in **C/C++** programming language that will compile and execute on *linprog.cs.fsu.edu*. You ***MUST NOT*** use any scanner generators, such as *lex* (or *flex*).

- You should submit your source as a single file to ***Canvas*** course website. For this submission, you ***MUST*** rename your single source file (cscan.c or cscan.cpp) to <*your last name*>.c or <*your last name*>.cpp before submission. Your last name must be in all lowercase letters with no spaces if you have a multi-word last name.

- Your code must compile (as submitted, without further modification) on *linprog* using the flags and compiler (gcc or g++) that is specified in the project documents in order to receive any points. Verify this yourself before you submit to avoid a poor grade. All testing compilation will be performed exactly as specified in the project documents.

- The due date for the project will be announced in ***Canvas*** course website. A grade of zero will be recorded for missed exams and late assignments unless prior arrangements are made. Assignments turned in after the due date, but by the beginning of the next schedule class will be penalized **10%**. Assignments will not be accepted that are more than one class period late.