COP5621/CIS4930 - Spring 2020
Assignment 1
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 for each type of token the name of the token and the number of occurrences (`number`, `ident`, `char`, `string`, or the lexeme for the remaining types of tokens) in descending order of the number of occurences. Ties should first be broken by the string length of the token name and then by lexical order. It writes invalid characters (with a diagnostic) or a sequence of characters that do not comprise a valid token (e.g. unclosed string) to standard error output with an appropriate comment instead of to 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, newline 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)*                      # identifier
      | quote char quote                            # char literal
      | ditto str* ditto                            # string
      | "(" | ")" | "," | "." | ":" | ";"          # 1 char tokens
      | "?" | "[" | "]" | "{" | "}" | "~"
      | "&&" | "||" | "++" | "--" | "->"            # 2 char only tokens
      | ("|" | "^" | "&" | "+" | "-" |              # 2 or 3 char tokens
         "%" | "*" | "/" | "=" | "!" |
         ">" | ">>" | "<" | "<<") ("=" | null)
```

Blanks, tabs, newlines, and comments (enclosed between "/*" and "*/" or from "//" to the end of the current line) 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: Joe Shmoe                     *
 *      Class: COP5621                       *
 * Assignment: Asg 1 (Implementing a C scanner) *
 *    Compile: "gcc -g -o asg1 asg1.c"       *
 *                                           *
 *********************************************/
```

You may implement this assignment in any programming language that will compile and execute on the machine *program*. You cannot use any scanner generators, such as *lex*. You should make the output of your solution exactly match the output produced by the executable */home/faculty/whalley/cop5621exec/cscan*. As with all assignments, you should e-mail your source file to *whalley@cs.fsu.edu* as an attachment before the beginning of class when the assignment is due. Please put COP5621 somewhere on the subject line of the message. The assignment is due on January 17.

```
/* Example Program */
main()
{
   printf("This is a test.\n");
}
```

The following output will appear when the above program is input to *cscan*.

```
main
(
)
{
printf
(
"This is a test.\n"
)
;
}
```

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