CSMC 4180 Language Translation
Scanner Program #2

Write a well commented structured C scanner program, not C++, which will identify the following character sequences and return the associated token:

Tokens:

Sequence	token
begin	BEGIN
end	END
read	READ
write	WRITE
if	IF
then	THEN
else	ELSE
endif	ENDIF
while	WHILE
endwhile	ENDWHILE
variable	ID
integer	INTLITERAL
false	FALSEOP
true	TRUEOP
null	NULLOP
(	LPAREN
)	RPAREN
;	SEMICOLON
,	COMMA
:=	ASSIGNOP
+	PLUSOP
-	MINUSOP
*	MULTOP
/	DIVOP
!	NOTOP
<	LESSOP
<=	LESSEQUALOP
>	GREATEROP
>=	GREATEREQUALOP
=	EQUALOP
<>	NOTEQUALOP
and	ANDOP
or	OROP
eof	SCANEOF
lexical error	ERROR

Comments are identified by – – and everything following the – – through the end of line will be ignored.

The Listing filename which is constructed from the output filename from program 1 will have all lines from the input file be copied to a line buffer. Each line within the listing file will have a line number followed by the line buffer.

Lexical errors will be identified in the listing file with an explanation. No line numbers are used for a lexical error other than identifying the line the lexical error occurred in.

The total number of lexical errors will be identified at the end of the listing file.

The output file will contain a table consisting of the numeric (enum) token followed by the token name and the token buffer (actual text that identified the token).

One Temp file will not be written to at this time but will be opened and have a message printed into it "The Temp" and then appended to the end of the output file at the end of the program within the wrap up routine. Since this Temp file is appended to the end of the output file, it will not exist at the end of the program.

The second Temp file will have a message printed into it "Loop Information" this Temp file will still have the code to delete it commented out in the wrap up module but for this program it will remain when the program has completed

The main function will open the files and initialize parameters in a startup routine, then the scanner will be continuously call receiving the token and processing the line buffer, listing file, and output file until the SCANEOF token is received.

The main function will use the token buffer, token, and the line buffer to build the output file and listing files.

DO NOT USE BREAKS, GOTO, OR EXIT.

Include the program name and number, course name and number, group number, group members names and email addresses at the top of the program.
Name the files appropriately like main.c, file_util.c, file.h, scanner.c, scanner.h, etc

For example the program:

```
begin                  -- a program
a:= BB & A;
end
```

will generate the listing file:

```
1       begin                  -- a program
2       a:= BB & A;
Error.   & not recognized in line 2.
3       end

1       Lexical Errors.
```

and the output file:

```
token number:    0     token type:     BEGIN          actual token: BEGIN
token number:    4     token type:     ID             actual token: A
token number:   10     token type:     ASSIGNOP       actual token: :=
token number:    4     token type:     ID             actual token: BB
token number:   14     token type:     ERROR          actual token: &
token number:    4     token type:     ID             actual token: A
token number:    8     token type:     SEMICOLON      actual token: ;
token number:    1     token type:     END            actual token: END
token number:   13     token type:     SCANEOF        actual token: EOF
```