CSC 460 Language Translation
Program 2
Scanner

Write the scanner which will identify the following character sequences and return the associated token:

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
eof	SCANEOF
lexical error	ERROR

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

All lines from the input file will be copied to a line buffer which will be written to the listing file with a line number added to the front of the line.

Lexical errors will be identified in the listing file with an explanation.

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

The Temp file will not be written to at this time but will be opened and appended to the end of the output file then deleted when the program is completed. For now print a message to Temp which will be appended to the output file i.e. "The Temp.". Comment out the deletion of the Temp file at this time.

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

The Listing filename will be constructed from the output filename.

The main function will open the files and continuously call the scanner receiving the token 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.
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.
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
```