Output Format for Lexical Analyzer

Your lexical analyzer should output each token identified from the inputted MINI-L program. Each token should appear on a separate line of output, and the tokens should appear in the output in the same order as they appear in the inputted MINI-L program. To facilitate grading, the tokens must be outputted in the format described in the table below.

There are two types of lexical errors that your lexical analyzer should catch. They are described below.

Note: for this phase of the project, even syntactically incorrect MINI-L programs may still be parsed successfully into a list of tokens. The next phase of this project is where syntax errors will be captured.

List of Tokens

The following table describes the different kinds of tokens that may be outputted by your lexical analyzer. *Comments* and *whitespace* should be ignored by your lexical analyzer (you should not output any tokens for these).

Lexical Pattern in the Inputted MINI-L Program	**Token that Should Be Outputted**
Reserved Words	
function	FUNCTION
beginparams	BEGIN_PARAMS
endparams	END_PARAMS
beginlocals	BEGIN_LOCALS
endlocals	END_LOCALS
beginbody	BEGIN_BODY
endbody	END_BODY
integer	INTEGER
array	ARRAY
of	OF
if	IF
then	THEN
endif	ENDIF
else	ELSE
while	WHILE
do	DO
for	FOR
beginloop	BEGINLOOP
endloop	ENDLOOP

continue	CONTINUE
read	READ
write	WRITE
and	AND
or	OR
not	NOT
true	TRUE
false	FALSE
return	RETURN
Arithmetic Operators	
-	SUB
+	ADD
*	MULT
/	DIV
%	MOD
Comparison Operators	
==	EQ
<>	NEQ
<	LT
>	GT
<=	LTE
>=	GTE
Identifiers and Numbers	
identifier (e.g., "aardvark", "BIG_PENGUIN", "fLaMInGo_17", "ot73r")	IDENT XXXX [where XXXX is the identifier itself]
number (e.g., "17", "101", "90210", "0", "8675309")	NUMBER XXXX [where XXXX is the number itself]
Other Special Symbols	
;	SEMICOLON
:	COLON
,	COMMA
(	L_PAREN
)	R_PAREN
[	L_SQUARE_BRACKET
]	R_SQUARE_BRACKET

:=	ASSIGN

Lexical Errors to Catch

Your lexical analyzer should catch two different types of lexical errors. If any such error is encountered during parsing of a MINI-L program, your lexical analyzer should terminate immediately after reporting the error message. The error message must include information about the line number and column position number within the line of the token associated with the error. The details are below.

Error Type 1: Unrecognized Symbol

Your lexical analyzer should report an error and terminate if an unrecognized symbol is encountered that is outside of a comment. For example, consider the following MINI-L function:

```
01. function test;
02. beginparams
03. endparams
04. beginlocals
05. n : integer;
06. endlocals
07. beginbody
08.    read n;
09.    n := n + 1?
10.    write n;
11. endbody
```

In the above program, the "?" symbol at line 5 (which is outside of a comment) is not defined in the MINI-L language. Thus, your lexical analyzer should output an "unrecognized symbol" error when it encounters the "?" (along with line number and position number information of the problematic symbol). For example:

```
Error at line 9, column 14: unrecognized symbol "?"
```

Error Type 2: Invalid Identifier

Your lexical analyzer should report an error and terminate if an invalid identifier is encountered. This can occur if the identifier starts with a digit or an underscore, or if the identifier ends with an underscore. For example, consider the following two MINI-L functions:

```
01. function test1;
02. beginparams
03. endparams
04. beginlocals
05. 2n : integer;
06. endlocals
07. beginbody
08. endbody
```

```
01. function test2;
02. beginparams
03. endparams
04. beginlocals
05. n_ : integer;
06. endlocals
07. beginprogram
08. endprogram
```

In both of the above functions, the identifier declared at line 5 is invalid. Thus, in both of these cases, your lexical analyzer should output an "invalid identifier" error when it encounters either the "2n" or the "n_". For example, in the first function above:

```
Error at line 5, column 0: identifier "2n" must begin with a letter
```

And in the second function above:

```
Error at line 5 , column 0: identifier "n_" cannot end with an underscore
```
