1) Comment handling

The lexer starts off in an <INITIAL> state. Comments are handed in a <COMMENT> state. The first /* token sends us into a <COMMENT> state. An int reference commNest keeps track of how nested the current token is in a comment. commNest starts at 0 and increments whenever a /* token is seen. It decrements whenever a */ token is seen, and if commNest is 0, we go back into an <INITIAL> state. Most tokens encountered during a <COMMENT> state is ignored with the exception of /*, */, and a newline, which requires us to update the linenumber and linepos.

2) String handling

When a double-quote " token is received from the <INITIAL> state, the lexer transitions into the <STRING> state. A *strBuffer* reference (initialized to the empty string at the end of previous string) tracks characters of this string. Lexer then stays in the <STRING> state and adds following tokens to the *strBuffer* reference, unless the token falls into one of the following 9 categories:

1) " (Double-quote) : Lexer transitions into <INITIAL> state, outputting STRING(<*strBuffer*>)
2) \ (Single Backslash) : Lexer transitions into <IGNORESEQ> state, ignoring all non-printable tokens (\n, \f, \t, " ") until another \ is received, in which case it returns to the <STRING> state. If printable characters are received from the <IGNORESEQ> state, it prints the error message *"illegal use of printable character from IGNORESEQ"*
3) \\ (Double Backslash) : Lexer adds the literal single backslash "\\" character to buffer for printing and remains in <STRING> state
4) "\"" (Escaped double-quote ") : Lexer adds the literal double quote "\"" character to buffer
5) "\\^". (Control character '\^.') : Lexer calls the *strFromCtrl(yytext)* method, which translates the control character into its printable version if it exists, or else prints it as is. If it is not a valid control sequence, the error message *"Illegal control character sequence. Ignoring."* is printed.
6) \ddd (Backslash followed by 3-digit ascii code) : The lexer prints the message *"Interpreting ascii character sequence <yytext>"* and if the 3-digits are in the range 0 - 255 inclusive, adds to the buffer the character corresponding to the ascii value. Else, it prints the error message "Invalid ascii value not between 0 to 255."
7) "\\n" | "\\t" | " " | "\\f" : The lexer adds the literal newline, tab, space or form-feed, respectively, to the *strBuffer.*
8) [0-9]+ : The lexer adds the digit as string literals to the buffer. This is already handled by the above logic but is added as a separate case just for clarity.

is one of " (double-quote), "\" (backslash)

3) Error handling

There are four main cases where we call ErrorMsg.error.

1. If invalid chars are found in strings (for specifics, see string section).
2. A fall through case at the end of tiger.lex when nothing else matches because the character is illegal.
3. If a comment is left unclosed at the end of a program.
4. If a string is left unclosed at the end of a program.
5. For string-specific errors, refer to *2) String handling* above.

4) End-of-file handling

The *eof()* function checks the value of the abovementioned two references, *inComment* and *inString*, to determine whether there are unclosed comments or strings at EOF. If so, it prints an error message. Regardless, it prints the position of the EOF from the start of the file.

5) Other Interesting Features

An invalid use of backslash (such as “\s” or “\k”) within a string would cause this lexer to ignore all subsequent tokens while printing the error message *“illegal use of printable character from IGNORESEQ”* until a closing backslash is entered.