The Lexical Analyzer should be designed as a procedure that will be called every time that a token is needed by the parser. The routine should have at least two parameters that are returned to the calling routine: Code which is the numeric representation of the token and String which is the character representation of the token.

An input line has 80 characters. The entire input line must be printed at one time. All characters are to be used. There are no reserved columns on an input line. New lines will terminate lexical entities except multi-line comments.

The Lexical Analyzer returns only lexigraphically correct entities and detects certain error conditions. Upon encountering such an error condition, the lexical analyzer prints an informative message and continues operation until a valid token is found and returned.

Multi-line comments are enclosed by /* and */. Comments may start and end anywhere and are not embedded. Single line comments start with // and end with a new line. Single line comments start with // and go to the end of the line.

Diagnostic print flags are used to control debugging aids and diagnostic information for the rest of the project. The flag token will be delimited by ## and is also skipped similar to the skipping of comments. A flag token begins with a + or – and is followed by an integer number. The integer number specifies which flag will be set. “+” means turn the flag on and “–” means turn the flag off. There may be more than one flag setting per flag token. All flag manipulation will be indicated at the beginning of the flag token. Flags have a global context. For example:

```plaintext
##+9-3 this turns on flag 9 and turns off flag 3 ##
```

The initial list of flags is:

- 0 - set(+) or clear(-) all flags
- 1 - print source lines
- 2 - print the lexical token and code that is returned

All flags will initially be turned off except flag 1, which will always be turned on. Flags will control all output produced by the project. Flags 0-31 are reserved. You may add other flags starting at flag 32.

A list of reserved words is given below:

- Reserve words: END, PROGRAM, DECLARE, SCALAR, VECTOR, MATRIX, REAL, INTEGER, PROCEDURE, VALUE, REFERENCE, EXECUTE, STDIN, STOUT, CALL, ELSE, IF, THEN, FOR, BY, UNTIL, DO

Identifiers start with a lower case letter and may contain up to 15 additional lower case letters, upper case letters. digits, and _s.

Integer values and real values have 32-bit accuracy. Integer values are represented in the source language as d^*, where the * represents a continuous sequence of 9 or less significant digits. Real values are represented as d^*.d^*, where the total number of significant digits does not exceed 7.

Strings start and stop with a “,” may contain any character other than a “,” and will be contained on one line.

Single ASCII characters: ;, :, [, ], (, ), <, >, !, +, -, *, /, {, }, &,
Electronic Reporting:

E-mail in one message the following to grossman@cs.clemson.edu with a subject line of “Lexical Analyzer”. Failure to use the correct subject line or follow the directions will result in a 10 point deduction for this milestone.

1. Attach in non-compressed mode your source program for this milestone.

Meeting:

You will have a 5 minute meeting with me to talk about your lexical analyzer. This meeting will take place by February 5.

Paper Reporting:

Print the header, with partnership names, e-mail addresses, date, and time. Failure to have the partner’s names, e-mail addresses, date, and time at the beginning of your output will result in a grade of 0 for that milestone.

Flag 1: Print complete source lines starting in column 1.

Flag 2: Print the returned token along with it numeric code. All debugging print should be to the right of the output page. The source lines and debugging print will be part of the same physical output. See sample output for ideas.

The following conditions must be recognized, and an informative message printed:

1. numeric constant containing too many significant digits
   a. integer constants that exceed 9 digits in length
      b. real constants that exceed 7 significant digits

2. invalid ascii character

3. invalid reserve word

4. invalid identifier
Discussion: