CSCI 380 Compiler EBNF

This is the syntax for the subset of Modula-2 that you implement.

```
EBNF Symbols:
    {}  contents repeated zero or more times
    []  contents present zero or one time
    |   read as 'or'; only one of the alternatives is present
Any symbol in " " is a terminal.
============================================================

program_module -->
MODULE  ident ; block ident .

block -->
{ declaration } BEGIN statementSequence END

declaration -->
  CONST { constantDeclaration ; }
| TYPE  { typeDeclaration ; }
| VAR   { variableDeclaration ; }
| procedureDeclaration ;

statementSequence -->
statement { ; statement }

constantDeclaration -->
ident = number

typeDeclaration -->
ident = type

type -->
  simpleType          | ident
| ARRAY "[" integer | ident .. integer | ident "]"
          OF simpleType

simpleType -->
  CARDINAL
| INTEGER
| REAL
```

variableDeclaration -->
identList **:** *type*

procedureDeclaration -->
procedureHeading **;** *block ident*

procedureHeading -->
PROCEDURE *ident* [*formalParameters*]

formalParameters -->
([*fpSection* { **;** *fpSection* }]) [**:** *simpleType*]

fpSection -->
[**VAR**] *identList* **:** *type*

identList -->
ident { **,** *ident* }

designator -->
ident ["[" *expression* "]"]

expList -->
expression { **,** *expression* }

expression -->
simpleExpression [*relOperator simpleExression*]

simpleExpression -->
[+ | -] *term* { *addOperator term* }

term -->
factor { *mulOperator factor* }

factor -->
  designator
| *ident* (*expList*)
| *number*
| (*expression*)
| **NOT** *factor*

```
statement -->
[   assignment            | procedureCall
      | ifStatement           | loopStatement
  | EXIT                  | RETURN expression
  | RDCARD ( ident )      | RDINT ( ident )
  | RDREAL ( ident )      | WRLN
  | WRCARD  expression    | WRREAL  expression
  | WRINT   expression
]     (*** note []; empty statement is possible ***)

assignment -->
designator := expression

procedureCall -->
ident [ ( [ expList ] ) ]

ifStatement -->
IF expression THEN statementSequence
[ ELSE statementSequence ] END

loopStatement -->
LOOP statementSequence END



/*********************************************************/
/* The following are NOT productions; Incorporate them directly
into other productions, or into the lexer */


relOperator -->
= | # |<>| <= | >= | < | >

addOperator -->
+ | - | OR

mulOperator -->
* | DIV | MOD | / | AND

ident -->
    letter { letter | digit }

letter --> A..Z | a..z | _
```

```
number -->
integer | real

integer -->
digit { digit }

/* The following are optional */

real -->
digit { digit } . { digit } [ scaleFactor ]

scaleFactor -->
E [ + | - ] digit { digit }
```