Statement: Considering a small programming language (that we shall call mini-langauge), you have to write a scanner (lexical analyzer)

Task 1: Minilanguage Specification
Deliverables:

1) Lexic.txt (file containing mini language lexic description; see example)
2) token.in (containing the list of all tokens corresponding to the minilangauge)
3) Syntax.in - the syntactical rules of the language

Task 2: Review the mini language specification of a colleague

The minilanguage can be a restricted form of a known programming language, and should contain the following:
- 2 simple data types and a user-defined type
- statements:
    - assignment
    - input/output
    - conditional
    - loop
- some conditions will be imposed on the way the identifiers and constants can be formed:
i) Identifiers: no more than 256 characters
ii) constants: corresponding to your types

Example: the minilanguage specification should include lexical and syntactical details:
Specification (file Lexic.txt)
Alphabet:
a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
b. Underline character '_';
c. Decimal digits (0-9);

1. **Lexic:**

a.Special symbols, representing:
- operators + - * / := < <= = >=
- separators [ ] { } : ; space
- reserved words:
  array char const do else if int of program read
  then var while write

b.identifiers
-a sequence of letters and digits, such that the first character is a letter; the rule is:
identifier ::= letter | letter{letter}{digit}
letter ::= "A" | "B" | . ..| "Z"
digit ::= "0" | "1" |...| "9"

c.constants

1.integer - rule:
   noconst:=+no|-no|no
   no:=digit{no}
2.character
   character:='letter'|'digit'
3.string
   constchar:="string"
   string:=char{string}
   char:=letter|digit

2. **Syntax:**

The words - predefined tokens are specified between " and ":

Sintactical rules: (file Syntax.in)

```
program ::= "VAR" decllist ";" cmpdstmt "."
decllist ::= declaration | declaration ";" decllist
declaration ::= IDENTIFIER ":" type
type1 ::= "BOOLEAN" | "CHAR" | "INTEGER" | "REAL"
arraydecl ::= "ARRAY" "[" nr "]" "OF" type1
type  ::= type1|arraydecl
cmpdstmt ::= "BEGIN" stmtlist "END"
stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= simplstmt | structstmt
simplstmt ::= assignstmt | iostmt
assignstmt ::= IDENTIFIER ":=" expression
expression ::= expression "+" term | term
term ::= term "*" factor | factor
factor ::= "(" expression ")" | IDENTIFIER
iostmt ::= "READ" | "WRITE" "(" IDENTIFIER ")"
structstmt ::= cmpdstmt | ifstmt | whilestmt
ifstmt ::= "IF" condition "THEN" stmt ["ELSE" stmt]
whilestmt ::= "WHILE" condition "DO" stmt
condition ::= expression RELATION expression
RELATION ::= "<" | "<=" | "=" | "<>" | ">=" | ">"
```