Language Specification:

1 . Language Definition:

1.1 Alphabet:

a. Upper (A-Z) and lower-case letters (a-z) of the English alphabet

b. Decimal digits (0-9);

Lexic:

**a.** Special symbols, representing:

- operators + - \* / <- < > == <> >= <=

- separators () [ ] { } : ; space

- reserved words:

array char string const do while else if then int program get function variable write

**b.** Identifiers

-a sequence of letters and digits, such that the first character is either a letter or a digit, but the digits can’t be followed by letters; the rule is:

<identifier> ::= <letter> | <digit > | <letter><more\_letters>| <digit><more\_digits>|<more\_letters><more\_digits>

<letter> ::= “a” | “b” | ...| “z” | “A” | “B” | ...| “Z”

<digit> ::= “0” | “1” |...| “9”

<more\_letters> ::= <letter>|<letter><more\_letters>

<more\_digits> ::= <digit> |<digit> <more\_digits>

**c.** Constants

1.integer :

<numberconst>::= +<number>|-<number>|<number>|”0”

<non\_zero\_digit> ::= "1"|"2"|....|"9"

<number> ::= <non\_zero\_digit> | <non\_zero\_digit><more\_digits>

2.character

<character> ::= <letter>|<digit>

3.string

<char> ::= <letter>|<digit>

<constchar> ::= <string>

<string> ::= <char> | <char><string>

4. boolean

<value> ::= “TRUE” | “FALSE”

2. Syntax:

a) Syntactical rules:

<program> ::= “VAR” <declarationlist> ";" <compoundStatement>"."

<declarationList> ::= <declaration> “;” | <declaration> “;” <declarationList>

<declaration> ::= <identifier> ":"< type>

<type> ::= <BOOLEAN>| <CHAR>| <INTEGER>

<arrayDeclaration> ::= “ARRAY” "[" <numberconst> "]" "OF" <type>

<types> ::= <type>|<arrayDeclaration>

<compoundStatement> ::= <START><statementList><END>

<statementList> ::= <statement> | <statement>";" <stmtlist>

<statement> ::= <simpleStatement> | <structStatement>

<simpleStatement> ::= <assignstmt> | <iostmt>

<assignstmt> ::= <identifier> "<-" <expression> “;”

<expression> ::= <expression><operation><term> |< term>

<operation> ::= “+” | “-” | “mod” | “div” | “\*”

<term> ::= <term><operation><factor> | <factor>

<factor> ::= "(" <expression> ")" | <identifier>

<iostmt> ::= "GET" "(" <identifier> ")" “;” | "WRITE" "(" <identifier> ")" “;”

< structStatement> ::= <compoundStatement> | <ifstmt> | <whilestmt>

<ifstmt> ::= "IF" <condition> "THEN" <statement> |"IF" <condition> "THEN" <statement> "ELSE" <statement>

<whilestmt>::= "WHILE" <condition> "DO" <statement>

<condition> ::= <expression><relation><expression>

<relation> ::= "<" | "==" | "<>" | ">" | “<=” | “>=”

**b)** Write a small program into your programming language (for example: find the first k prime numbers, print all prime numbers less than k)

Program : sum of digits of given number k

VAR

K : INTEGER ;

D : INTEGER ;

S : INTEGER ;

X : INTEGER ;

START

S <- 0 ;

GET (K) ;

X <- K ;

WHILE K>0 DO

START

D <- X mod 10 ;

S <- S + D ;

X <- X div 10 ;

END

WRITE (S) ;

END