Language Specification:

1. Language Definition:

1.1 Alphabet:

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

b. Underline character '\_';

c. Decimal digits (0-9);

Lexicon:

a. Special symbols, representing:

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

- separators [ ] { } ; space

- reserved words:

read write list const do while if else integer character

b. identifiers

-a sequence of letters and digits, such that the first character is a letter; the rule is:

identifier ::= letter {letter | digit}

letter ::= "a" | "b" | ...| "z" | "A" | "B" | ...| "Z"

digit ::= "0" | "1" |...| "9"

nonzero\_digit ::= | "1" |...| "9"

c. constants

1.integer - rule:

intconst:=+ digits |- digits | digits

digits:=digit | nonzero\_digit{digit}

2.character

character:=letter | digit | ’,’ | ’?’ | ’!’ | ’:’ | ‘.’ | ‘(‘ | ‘)’ | ‘{‘ | ‘}’ | ‘[‘ | ‘]’ | ‘%’

3.string

constcharacter:="string"

string:= character {character}

character:=letter | digit | ’,’ | ’?’ | ’!’ | ’:’ | ‘.’ | ‘(‘ | ‘)’ | ‘{‘ | ‘}’ | ‘[‘ | ‘]’ | ‘%’

2.2 Syntax:

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

a) Syntactical rules:

program ::= "vars" decllist ";" cmpdstmt "."

decllist ::= declaration | {declaration}

declaration ::= type “ “ list\_identifiers “;”

list\_identifiers ::== identifier { ‘,’ identifier }

type1 ::= "char" | "integer"

arraydecl ::= "list" "[" nr "]" type1 identifier “;”

type ::= type1|arraydecl

cmpdstmt ::= "START" stmtlist "STOP"

stmtlist ::= stmt | stmt ";" stmtlist

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt

assignstmt ::= IDENTIFIER "=" expression

expression ::= expression "+" term | expression "-" term | term

term ::= term "\*" factor | term "/" factor | term "%” factor | | factor

factor ::= "(" expression ")" | IDENTIFIER

iostmt ::= "read" | "write" "("identifier")"

structstmt ::= cmpdstmt | ifstmt | whilestmt

ifstmt ::= "if" “(“ condition “)” “{“ stmt “}” {“else” “{“ stmt “}” }

whilestmt ::= "while" “(“ condition “)” "{" stmt “}”

condition ::= expression RELATION expression

RELATION ::= "<" | "<=" | "==" | "!=" | ">=" | ">"