Alphabet:

1. Letters from the following sets:

* [A-Z] , belonging to English alphabet
* [a-z], belonging to English alphabet

1. Special characters:

* ‘\_’ (underline)

1. Decimal digits belonging to the following set: [0,9]
2. Lexic:
3. Special symbols:

* Operators: + , - , \* , / , := , < , <=, = , =>, >
* Separators: : , ; , space , [ ]
* Reserved words: list,char,const,let, while, if, else, int, input, print, for, string

1. Identifiers:

* sequence of letters,digits and allowed special characters, s.t. the first char is a letter;

Identifier:= letter | letter{allowed\_identifier\_characters}

letter:= “A” | “B” | … | “Z” | “a” | “b” | … | “z”

digit:= “0” | “1” | … | “9”

allowed\_identifier = letter | digit | “\_”

1. Constants
2. Integer –rule:

nrConst:=[”+”]no|”-”no

no:=digit{no}

1. Character:

character:=letter|digit

1. String:

string:= ‘charArr’

charArr:=char{charArr}

char:= letter | digit

1. Syntax:

Reserved words are specified between double quotes:

* program:= “DECL” “{” declList ”}” “PROG” cmpdStmt
* declList:= declaration | declaration declList
* declaration:= type IDENTIFIER”;” | type IDENTIFIER “=” expression “;”

| arrayDecl = “[” [{expression [“,”] }] “]”

* type:= “BOOL” | “CHAR” | “INT” | “FLOAT” | arrayDecl
* arrayDecl:= “LIST<”type”>[”no”]”
* cmpdStmt:= type “{” stmtList “}”
* stmtList := stmt | stmt “;” stmtList
* stmt:= simpleStmt | complexStmt
* simpleStmt := assignStmt | ioStmt (| returnStmt????)
* assignStmt:= IDENTIFIER “=” expression
* expression = mathExpression | stringExpression
* mathExpression:= mathExpression “+”|”-”|”\*”|”/” factor | factor
* factor:= “(”mathExpression”)” | IDENTIFIER | no
* stringExpression:= stringExpression “+” string | string
* iostmt:= “INPUT” “(” IDENTIFIER ”)” | “PRINT” “(” expression | IDENTIFIER “)”
* complexStmt:= stmtList | ifStmt | whileStmt | forStmt
* ifStmt:= “IF” “(” condition “)” “{“ stmtList “}” [“ELSE” “{” stmtList “}”]
* whileStmt:= “WHILE” “(“ condition ”)” “{” stmtList “}”
* forStmt:=”for” “(” LET assignStmt “;” condition “;” assignStmt “)” “{” stmtList “}”
* condition:= expression RELATION expression | expression
* RELATION := “<” | “<=” | “==” | “!=” | “>=” | “>”