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);

Lexic:

a.Special symbols, representing:

- operators:

= + - \* / % == != > < >= <= ++ -- ~ & | ^ << >> ! && ||

- separators [ ] { } ; : space \n \t , “

- reserved words:

begin end def\_char def\_short def\_int def\_long def\_float def\_double def\_void break const continue do else if loop aslongas in out sqrt

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" | "a" | "b" | ... | "z"

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

non\_zero\_digit = "1" |...| "9"

zero\_digit = "0"

sign = ["+" | "-"]

dot = “.”

c.constants

1.a. integer:

integer = zero\_digit | non\_zero\_digit {digit}

3. string:

character = letter | digit

string::=”{character}”

4. float:

float = integer [dot digit {digit}]

2.2 Syntax:

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

a) Sintactical rules:

program = “begin” statementlist “end”

statementlist = statement;{statement}

statement = vardecl | simplestatement | structstatement | “end”

vardecl = type identifier

type = primarydecl | arraydecl

primarydecl = “def\_char” | “def\_short” | “def\_int” | "def\_float" | "def\_double"

arraydecl = primarydecl identifier”[“ unsigned\_integer“]”

positive\_number = non\_zero\_digit {digit}

compstatment = “{“ statementlist”}”

simplestmt = assignment | outstatement | instatement

assignment = identifier “=” expression

expression = (term | expression operation expression | “(” expression operation expression “)”)

operation = “+” | “-” | “\*” | “/” | “%” | “^” | "&" |

term = identifier | integer | float | string

outstatement = "out" "<<" identifier

instatement = “in” “>>” identifier

structstatement = ifstatement | whilestatement | forstatement | compstatment

ifstatement = “if” “(“ condition “)” statement {else statement }

condition = expression RELATION expression

whilestatement = “aslongas” “(“ condition “)” compstatment

forstatement = “loop” “(“ assignment; condition; [assignment]“)” statement

RELATION = "<" | "<=" | "==" | "!=" | ">=" | ">" | "and" | "or"

First:

begin  
def\_int a, b, c;  
in>>a>>b>>c;  
def\_int delta;  
delta=b\*b-4\*a\*c;  
def\_float x1,x2;  
x1=((-b)-sqrt(delta))/(2 \* a);  
x2=((-b)+sqrt(delta))/(2 \* a);  
out<<x1<<" "<<x2;  
end

Second:

begin  
def\_int a, b, c;  
in >> a >> b >> c;  
if((a > b) && (a > c))  
{  
 out << a;  
 end  
}  
if((b > a) && (b > c))  
{  
 out << b;  
 end  
}  
if((c > a) && (c > b))  
{  
 out << c;  
 end  
}  
end

Third:

begin  
def\_int arr[10];  
def\_int n;  
in >> n;  
def\_int sum;  
sum = 0;  
loop (def\_int i = 0; i < n; i = i + 1)  
{  
 in >> arr[i];   
}  
loop (def\_int i = 0; i < n; i = i + 1)  
{  
 sum = sum + arr[i];  
}  
out << sum;  
end