**Lexic**

Alphabet:

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

- Decimal Digits (0-9)

a.

- operators: inc, dec, =, ==, >=, <=, \*, /, mod, +, -, <, >

- separators: [] {} : ; space < > ( )

- reserved words:

number, decimal, str, car, let, seq, loop, vector, fi, fie, fiefi, $<<, $>>, rad, var

b.

identifier := letter | letter{letter}{digit}

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

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

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

number rule:

number\_const := ("+" | "-") number | number

number := "0" | non\_zero\_digit{digit}

deciaml rule:

decimal\_const := ("+" | "-")decimal | decimal

decimal := number "." number

car rule:

car := letter | digit

string rule:

str := {car}

**Syntax**

program := {statement}

statement := simple\_statemnt {statement} | simple\_statemnt

simple\_statemnt := declaration | assignment | if | ifelse | while | for | io | inc | dec

declaration := simple\_declaration | declaration\_assignment

simple\_declaration := "var" identifier ":" type ";"

declaration\_assignment := "var" identifier ":" type "=" expression ";"

assignment := identifier "=" expression ";"

simple\_type := "number" | "decimal" | "str" | "car"

type := simple\_type | "vector<" simple\_type ">"

identifier := letter | letter {letter} {digit}

constant := number\_const | decimal\_const | str\_const | car\_const

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

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

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

symbols := " " | "\n"

car := letter | digit | symbols

str := {car}

number\_const := ("+" | "-") number\_const\_simple | number\_const\_simple

number\_const\_simple := non\_zero\_digit {digit} | "0"

decimal\_const := ("+" | "-") decimal\_const\_simple | decimal\_const\_simple

decimal\_const\_simple := digit {digit} "." digit {digit}

str\_const := `{car}`

car\_const := "'" car "'"

if := "fi" "(" condition ")" "{" {statement} "}" ["else" "{" {statement} "}"]

ifelse := "fi" "(" condition ")" "{" {statement} "}" {("fiefi" "(" condition ")" "{" {statement} "}")} ["else" "{" {statement} "}"]

while := "loop" "(" condition ")" "{" {statement} "}"

for := "let" identifier "in" "seq(" sequence ") {" {statement} "}"

sequence := start "," end "," step | start "," end | end

start := expression | identifier | constant

end := expression | identifier | constant

step := expression | identifier | constant

read := $"<<" identifier ";"

print := "$>>" expression ";" | "$>>" identifier ";" | "$>>" constant ";"

io := read | print

condition := expression relation expression

relation := "<" | ">" | "<=" | ">=" | "==" | "!="

expression := normal\_expression | sqrt\_expression

normal\_expression := normal\_expression {("+" | "-") term} | term

term := term {("\*" | "/" | "mod") factor} | factor

factor := "(" normal\_expression ")" | identifier | number\_const | decimal\_const

sqrt\_expression := "rad" "(" (identifier | number\_const\_simple | decimal\_const\_simple) ")"

inc := "inc" identifier ";"

dec := "dec" identifier ";"

**Token**

inc

dec

=

==

>=

<=

\*

/

mod

+

-

<

>

[

]

{

}

(

)

:

;

space

number

decimal

str

car

let

seq

loop

vector

fi

fie

fiefi

$<<

$>>

rad

var