Lab1 B

Minilanguage Specification

Alphabet:
a) A-Z,a-z (upper and lowercase letters of the English Alphabet)
b) 0-9 (digits)
c) _ (underline character)

1. Lexic:

a) Special symbols, representing:
- operators:
  - +, -, *, / (arithmetic)
  - is (==) , smallerEq (<=), smaller (<), greater(>), greaterEq (>=), isNot (!=) (relational)
  - takes (assignment)

- separators: '(' , ')' , '[' , ']' , '{' , '}' , ':' , ';' , ',' , ' ' -> (space)
- reserved words: read, write, if, elif, else, for, while, int, float, string, char, return, start, end, array

b) Identifiers

A sequence of letters, digits or "_" such that the first character is "_" or a letter

identifier = (letter | "_") {letter | digit | "_"}
letter = "A" | "B" | "D" | ... | "Z" | "a" | "b" | ... | "z"
digit = "0" | non_zero_digit
non_zero_digit = "1" | "2" | ... | "9"

c) Constants:

int = "0" | ["+" | "-"] non_zero_digit {digit}
char = letter | digit
string = {char}

char_const = "'" char "'"
string_const = """ {char} """
int_const = "0" | ["+" | "-"] non_zero_digit {digit}

Syntax:

```
program ::= "start" compound_statement “end”
statement ::= (declaration | assignment_statement | if_statement |
          while_statement | return_statement | for_statement | iostmt)

statement_list ::= statement | statement ";" statement_list
compound_statement ::= "{" statement_list "}"

expression ::= expression + term | expression - term | term
term ::= term * factor | term / factor | factor
factor ::= "(" expression ")" | IDENTIFIER | CONST

iostmt ::= "read" "(" IDENTIFIER ")" | "write" "(" IDENFITIER ")" | "write "(" CONST ")"

simple_type ::= "int" | "string" | "char"
array_declaration ::= "array" " " simple_type " " IDENTIFIER "[" "]"
declaration_stmt ::= simple_type " " IDENTIFIER | array_declaration

assignment_statement ::= IDENTIFIER "=" expression
if_statement ::= "if" "(" condition ")" compound_statement | "if" "(" condition
")"compound_statement "else" compound_stawtement
while_statement ::= "while "(" condition ")" compound_statement
return statement ::= "return expression
for_statement ::= "for" for_header compound_statement
for_header ::= "(" "int" assignment_statement ";" condition ";" assignment_statement ")"

condition ::= expression relation expression
relation ::= "smaller" | "smallerEq" | "is" | "isNot" | "greaterEq" | "greater"
```

Recognized Tokens:
int
float
string
read
write
if
elif
else
while
takes

smallerEq
smaller
greaterEq
greater
is
isNot
+
-
*
/
(
)
{
}
,
;
:
"
'
_
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
a
b

c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
0
1
2
3
4
5
6
7
8
9