Alphabet:

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

b. Underline character '_';

c. Decimal digits (0-9);

1. Lexic:

a. Special symbols:

- operators:

```
+ - * / = < <= == != >= > && || !
```

- separators:

```
[ ] { } : ; space
```

- reserved words:

```
let const
char i32 bool
if else while
print read
```

b. identifiers:

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

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

c. constants

1. integer:

```
const_int ::= [("+" | "-")] const_uint
const_uint ::= digit {digit}
```

2. character:

```
const_char ::= "'" char "'"
```

```
      char ::= letter | digit | symbol
      symbol ::= " " | "_"

    3. string:
      const_string ::= '"' string '"'
      string ::= {char}

    4. boolean:
      const_bool ::= "true" | "false"

2. Syntax:
  program ::= statement_list

  statement_list ::= statement
    | statement statement_list
  statement ::= declare_statement
    | expression_statement
    | ";"
  declare_statement ::= "let" identifier ":" type ";"
  expression_statement ::= expression ";"
  const ::= const_bool
    | const_char
    | const_string
    | const_int

  expression ::= const
    | block_expression
    | assign_expression
    | if_expression
    | while_expression
    | print_expression
    | read_expression
    | unary_operator_expression
    | binary_operator_expression
    | group_expression
  block_expression ::= "{" block_content "}"
  block_content ::= statement_list | statement_list expression
  assign_expression ::= identifier "=" expression
  if_expression ::= "if" "(" expression ")" block_expression
  while_expression ::= "while" "(" expression ")" block_expression
  print_expression ::= "print" "(" expression ")"
  read_expression ::= "read"

  unary_operator_expression ::= unary_operator expression
  unary_operator ::= "-" | "!"
  binary_operator_expression ::= expression binary_operator expression
  binary_operator ::= arithmetic_operator
    | booolean_operator
    | comparison_operator
  arithmetic_operator ::= "+" | "-" | "*" | "/" | "%"
  booolean_operator ::= "&&" | "||"
  comparison_operator ::= "==" | "!=" | "<" | ">" | "<=" | ">="
  group_expression ::= "(" expression ")"

  type ::= basic_type | array_type
  basic_type ::= "i32" | "char" | "bool"
  array_type ::= "[" type ";" const_uint "]"
```