--- Lexic.txt

```
a. Special symbols:
   - arithmetic operators + - * / ^
   - comparison operators < > <= >= == !=
   - separators [ ] { } : space -> < >
   - reserved words:
       if else while funk char bool int program input output mut
imm borrow pack

b. Identifiers
   identifier ::= letter [{letter} {digit}]
   letter ::= "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
   digit ::= "0" | "1" |...| "9"

c. Constants
   1. integer:
       int_constant := [sign] int
       int := digit{int} // we can allow integer constants to be
written with leading zeroes for padding reasons
       sign := "+" | "-"

   2. character
       character_constant := 'char'
       char := letter|digit

   3.string
       string_constant := "string"
       string := char {string}
```

--- Syntax.in

```
program := ["pack" package_name] [import_list] statement_list

statement_list := statement ["\n" statement_list]
statement := simple_statement | struct_statement

simple_statement := declaration | assignment_statement |
io_statement
struct_statement := block_statement | if_statement |
while_statement

declaration := ("imm" | "mut") IDENTIFIER ":" type
```

```
assignment_statement := IDENTIFIER "=" expression
io_statement := "input" | "output" "(" IDENTIFIER ")"

block_statement := "{" statement_list "}"
if_statement := "if" condition "->" statement ["else ->"
statement]
while_statement := "while" condition "->" statement

type := primitive[array_declaration] | (class_name ["<"
class_list ">"])
primitive := "bool" | "char" | "int" | "real"
class_declaration := "class" class_name [generic]
generic := "<" generic_parameter_list ">"
generic_parameter_list := generic_parameter ","
generic_parameter_list]
generic_parameter := IDENTIFIER ":" class_name
class_list := class_name [',' class_list]
array_declaration := "[" nr "]"
class_name := letter [{letter}]
letter := "a" | ... | "z" | "A" | ... | "Z"

expression := expression arithmetic_operator term | "("
expression ")" [arithmetic_operator term]
term := IDENTIFIER | expression
arithmetic_operator := "+" | "-" | "*" | "/" | "^"

package := [prefix "."] package_name
package_name := IDENTIFIER
import_statement := "!borrow" package_name
import_list := import_statement [import_list]




condition := expression RELATION expression
```

```
RELATION := "<" | "<=" | "=" | "<>" | ">=" | ">"
```