EBNF for the `smpl` Programming Language

letter = “a” | “b” | … | “z”.
digit = “0” | “1” | … | “9”.
relOp = “==“ | “!=“ | “<“ | “<=“ | “>“ | “>=“.

ident = letter {letter | digit}.
number = digit {digit}.

designator = ident{ "[" expression "]" }.
factor = designator | number | “**(**“ expression “**)**” | funcCall[1] .
term = factor { (“*” | “/”) factor}.
expression = term {(“+” | “-”) term}.
relation = expression relOp expression .

assignment = “**let**” designator “**<-**” expression.
funcCall = “**call**” ident [[2] “**(**“ [expression { “**,**” expression }] “**)**”].
ifStatement = “**if**” relation “**then**” statSequence [“**else**” statSequence] “**fi**”.
whileStatement = “**while**” relation “**do**” StatSequence “**od**”.
returnStatement = “**return**” [expression] .

statement = assignment | funcCall[3] | ifStatement | whileStatement | returnStatement.
statSequence = statement { “**;**” statement } [“**;**”][4] .

typeDecl = “**var**” | “**array**” “[“ number “]” { “[“ number “]” }.
varDecl = typeDecl indent { “**,**” ident } “**;**” .
funcDecl = [“**void**”] “**function**” ident formalParam “**;**” funcBody “**;**” .
formalParam = “**(**“ [ident { “**,**” ident }] “**)**” .
funcBody = { varDecl } “**{**” [statSequence] “**}**”.
computation = “**main**” { varDecl } { funcDecl } “**{**” statSequence “**}**” “**.**” .

Predefined Functions

InputNum()	read a number from the standard input
OutputNum(x)	write a number to the standard output
OutputNewLine()	write a carriage return to the standard output

---

[1] only non-void functions can be used in expressions, for example `y <- call f(x) + 1;`
[2] functions without parameters can be called with or without parantheses
[3] only void functions can be used in statements, e.g. `call do(); call this(x); call do;`
[4] the semicolon is a statement separator; non-strictly necessary terminating semicolons are optional