Go Grammar

program -> declarationList

declarationList -> declarationList declaration | declaration

declaration -> varDeclaration | funcDeclaration

---

varDeclaration -> **var** varDeclList typeSpecifier **\n**

varDeclList -> varDeclList, varDeclInitialise | varDeclInitialise

varDeclInitialise -> varDeclId | varDeclId = simpleExpression

varDeclId -> **ID** | **ID**[**NUMCONST**]

typeSpecifier -> int | bool | string

---

funcDeclaration -> **func ID**(params) typeSpecifier statementList **\n**

params -> paramList | **e**

paramList -> paramList , param

param -> paramIds typeSpecifier

paramIds -> paramIds , **ID** | **ID**

---

statementList -> statementList statement | **e**

statement -> expressionStmt | selectionStmt | forStmt | returnStmt

expressionStmt -> expression **\n** | **\n**

selectionStmt -> **if** simpleExpression { statementList } elseIfList
| **if** simpleExpression { statementList } elseIfList **else** { statementList }

elseIfList -> elseIfList **else if** simpleExpression { statementList } | **e**

forStmt -> **for** varDecId **:=** simpleExpression **;** simpleExpression **;** simpleExpression **;** { statementList }

returnStmt -> **return** expression

---

expression -> mutable = expression | mutable += expression | mutable -= expression | mutable *= expression |
mutable /= expression | mutable++ | mutable– | simpleExpression

simpleExpression -> simpleExpression || andExpression | andExpression

andExpression -> andExpression **&&** unaryRelExpression | unaryRelExpression

unaryRelExpression -> **!** unaryRelExpression | relExpression

relExpression -> sumExpression relOperator sumExpression | sumExpression

relOperator -> == | **!=** | < | <= | > | >=

sumExpression -> sumExpression sumOperator mulExpression | mulExpression

sumOperator -> + | **-**

mulExpression -> mulExpression mulOperator unaryExpression | unaryExpression

mulOperator -> ***** | / | %

unaryExpression -> unaryOperator unaryExpression | factor

unaryOperator -> + | **-**

factor -> immutable | mutable

mutable -> **ID** | **ID**[expression]

immutable -> (expression) | functionCall | constant

functionCall -> **ID**(args)

args -> argList | **e**

argList -> argList , expression | expression

constant -> **NUMCONST** || **CHARCONST** || **STRINGCONST** || true || false