BNF Grammar

start	→ **PROGRAM IDENT ;** *varDec compStmt* **.**
varDec	→ **VAR** *varDecList*
	| ε
varDecList	→ *varDecList identListType* **;**
	| *identListType* **;**
identListType	→ *identList* **:** *type*
identList	→ *identList* **, IDENT**
	| **IDENT**
type	→ *simpleType*
	| **ARRAY [NUM .. NUM] OF** *simpleType*
simpleType	→ **INTEGER**
	| **REAL**
	| **BOOLEAN**
compStmt	→ **BEGIN** *stmtList* **END**
stmtList	→ *stmtList* **;** *statement*
	| *statement*
statement	→ *assignStmt*
	| *compStmt*
	| *ifStmt*
	| *whileStmt*
assignStmt	→ **IDENT :=** *expr*
	| **IDENT** *index* **:=** *expr*
index	→ **[** *expr* **]**
	| **[** *expr* **..** *expr* **]**
ifStmt	→ **IF** *expr* **THEN** *statement elsePart*
elsePart	→ **ELSE** *statement*
	| ε
whileStmt	→ **WHILE** *expr* **DO** *statement*
forStmt	→ **FOR IDENT** := *expr toPart expr* **DO** *statement*
toPart	→ **TO** | **DOWNTO**
exprList	→ *exprList* **,** *expr*
	| *expr*
expr	→ *simpleExpr relOp simpleExpr*
	| *simpleExpr*
*simpleExp*r	→ *simpleExpr addOp term*

| *term*

term → *term mulOp factor*
| *factor*

factor → **NUM**
| **FALSE**
| **TRUE**
| **IDENT**
| **IDENT** *index*
| **NOT** *factor*
| **–** *factor*
| **(** *exp* **)**

relOp → **<** | **<=** | **>** | **>=** | **=** | **<>**

addOp → **+** | **–** | **OR**

mulOp → ***** | **/** | **DIV** | **MOD** | **AND**