

## FIRST - SETS

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Start → PROGRAM IDENT ; [ VAR varDecList ] CompStmt .	First(start) = {PROGRAM}				
VarDecList → identListType {; identListType}			First(VarDecList) = First(IdenListType) = { IDENT }		
IdentListType → identList : type		First(IdenListType) = First(IdenList) = { IDENT }			
IdentList → IDENT {, IDENT}	First(IdenList) = { IDENT }				
Type → simpleType		First(Type) = {ARRAY, INTEGER, REAL, STRING}			
Type → ARRAY '[' NUM .. NUM ''] OF simpleType	First(Type) = {ARRAY}				
SimpleType → INTEGER	First(SimpleType)={INTE GER}				
SimpleType → REAL	First(SimpleType)={INTE GER, REAL}				
SimpleType → STRING	First(SimpleType)={INTE GER, REAL, STRING}				
CompStmt → BEGIN stmtList END	First(CompStmt) = {BEGIN}				
StmtList → statement {; statement}			First(StmtList) = First(Statement) = {BEGIN, READ, WRITE , IDENT, IF, WHILE, FOR}		
Statement → assignStmt		First(Statement) = {BEGIN, READ, WRITE , IDENT}			

Statement → compStmt	First(Statement) = {BEGIN}				
Statement → ifStmt		First(Statement) = {BEGIN, READ, WRITE, IDENT, IF}			
Statement → whileStmt		First(Statement) = {BEGIN, READ, WRITE, IDENT, IF, WHILE}			
Statement → forStmt		First(Statement) = {BEGIN, READ, WRITE, IDENT, IF, WHILE, FOR}			
Statement → READ '(' exprList ')'	First(Statement) = {BEGIN, READ}				
Statement → WRITE '(' exprList ')'	First(Statement) = {BEGIN, READ, WRITE}				
AssignStmt → IDENT [ index ] := expr	First(AssignStmt) = {IDENT}				
Index → '[' simpleExpr [.. simpleExpr] ']'	First(Index) = {}				
IfStmt → IF expr THEN statement [ ELSE statement ]	First(IfStmt) = {IF}				
WhileStmt → WHILE expr DO statement	First(WhileStmt) = {WHILE}				
ForStmt → FOR IDENT ':=' expr toPart expr DO statement	First(ForStmt) = {FOR}				
ToPart → TO	First(ToPart) = {TO}				
ToPart → DOWNT0	First(ToPart) = {TO, DOWNT0}				
ExprList → expr {' expr}					First(ExprList) = First(Expr) = { NUM, STRING, FALSE,

					TRUE, IDENT, NOT, -, ( }
Expr $\rightarrow$ simpleExpr {relOp simpleExpr}				First(Expr) = First(SimpleExpr) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( }	
SimpleExpr $\rightarrow$ term {addOp term}			First(SimpleExpr) = First(Term) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( }		
Term $\rightarrow$ factor {mulOp factor}		First(Term) = First(Factor) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( }			
Factor $\rightarrow$ NUM   STRING   FALSE   TRUE   IDENT [ index ]   NOT factor   '-' factor   '(' exp ')'	First(Factor) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( }				
RelOp $\rightarrow$ <   <=   >   >=   =   <>	First(RelOp) = {<, <=, >, >=, =, <>}				
AddOp $\rightarrow$ +   -   OR	First(AddOp) = {+, -, OR}				
MulOp $\rightarrow$ *   /   DIV   MOD   AND	First(MulOp) = {*, /, DIV, MOD, AND}				

## FOLLOW - SETS

	Iteration 1	Iteration 2
Start $\rightarrow$ PROGRAM IDENT ; [ VAR varDecList ] CompStmt .	Follow(Start) = { \$ }, Follow(varDecList) = First(CompStmt) = { BEGIN }, Follow(CompStmt) = { . }	
VarDecList $\rightarrow$ identListType {; identListType}	Follow(identListType) = { ; } Follow(identListType) += Follow(VarDecList) = { ; , BEGIN }	

IdentListType $\rightarrow$ identList : type	Follow(identList) = {:,} Follow(type) = Follow(IdentListType) = { ; , BEGIN }	
IdentList $\rightarrow$ IDENT {, IDENT}		
Type $\rightarrow$ simpleType	Follow(simpleType) = Follow(type) = { ; , BEGIN }	
Type $\rightarrow$ ARRAY '[' NUM .. NUM ']' OF simpleType	Follow(simpleType) = Follow(type) = { ; , BEGIN }	
SimpleType $\rightarrow$ INTEGER		
SimpleType $\rightarrow$ REAL		
SimpleType $\rightarrow$ BOOLEAN		
CompStmt $\rightarrow$ BEGIN stmtList END	Follow(stmtList) = {END}	
StmtList $\rightarrow$ statement {; statement}	Follow(Statement) = { ; } Follow(Statement) += Follow(stmtList) = { ; , END }	
Statement $\rightarrow$ assignStmt	Follow(assignStmt) = Follow(Statement) = { ; , END }	Follow(assignStmt) = Follow(statement) = { ; , END, ELSE }
Statement $\rightarrow$ compStmt	Follow(compStmt) = Follow(Statement) = { ; , END }	Follow(compStmt) = Follow(statement) = { ; , END, ELSE }
Statement $\rightarrow$ ifStmt	Follow(ifStmt) = Follow(Statement) = { ; , END }	Follow(ifStmt) = Follow(statement) = { ; , END, ELSE }
Statement $\rightarrow$ whileStmt	Follow(whileStmt) = Follow(Statement) = { ; , END }	Follow(whileStmt) = Follow(statement) = { ; , END, ELSE }
Statement $\rightarrow$ forStmt	Follow(forStmt) = Follow(Statement) = { ; , END }	Follow(forStmt) = Follow(statement) = { ; , END, ELSE }
Statement $\rightarrow$ READ '(' exprList ')'	Follow(exprList) = { } }	
Statement $\rightarrow$ WRITE '(' exprList ')'	Follow(exprList) = { } }	
AssignStmt $\rightarrow$ IDENT [ index ] := expr	Follow(index) = { : } Follow(expr) = Follow(assignStmt) = { ; , END }	Follow(expr) += Follow(assignStmt) = { ELSE, ; , END, THEN, DO, TO, DOWNT0, ', ' }
Index $\rightarrow$ '[' simpleExpr [.. simpleExpr] ']'	Follow(simpleExpr) = { . } Follow(simpleExpr) = { . , }	
IfStmt $\rightarrow$ IF expr THEN statement [ ELSE statement ]	Follow(expr) += THEN = { ; , END, THEN } Follow(statement) += ELSE = { ; , END, ELSE } Follow(statement) += Follow(IfStmt) = { ; , END, ELSE }	Follow(statement) += Follow(IfStmt) = { ; , END, ELSE }
WhileStmt $\rightarrow$ WHILE expr DO statement	Follow(expr) += DO = { ; , END, THEN, DO }	Follow(statement) += Follow(WhileStmt) = { ; , END, ELSE }

	Follow(statement) += Follow(WhileStmt) = { ; , END, ELSE }	
ForStmt → FOR IDENT ':'= expr toPart expr DO statement	Follow(expr) += First(ToPart) = { ; , END, THEN, DO, TO, DOWNT0 } Follow(toPart) = First(expr) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( } Follow(expr) += DO = { ; , END, THEN, DO, TO, DOWNT0 } Follow(statement) += Follow(ForStmt) = { ; , END, ELSE }	Follow(statement) += Follow(ForStmt) = { ; , END, ELSE }
ToPart → TO		
ToPart → DOWNT0		
ExprList → expr {' expr }	Follow(expr) += ' = { ; , END, THEN, DO, TO, DOWNT0, ' } Follow(expr) += Follow(ExprList) = { ; , END, THEN, DO, TO, DOWNT0, ' , ) }	
Expr → simpleExpr {relOp simpleExpr }	Follow(simpleExpr) += First(relOp) = { . , ], < , <= , > , >= , = , <> } Follow(relOp) = First(simpleExpr) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( } Follow(simpleExpr) += Follow(expr) = { . , ], < , <= , > , >= , = , <> , ; , END, THEN, DO, TO, DOWNT0, ' , ) }	Follow(simpleExpr) += Follow(expr) = { ELSE, . , ], < , <= , > , >= , = , <> , ; , END, THEN, DO, TO, DOWNT0, ' , ) }
SimpleExpr → term {addOp term }	Follow(term) = First(addOp) = { + , - , OR } Follow(addOp) = First(term) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( } Follow(term) += Follow(SimpleExpr) = { + , - , OR, . , ], < , <= , > , >= , = , <> , ; , END, THEN, DO, TO, DOWNT0, ' , ) }	Follow(term) += Follow(SimpleExpr) = { ELSE, + , - , OR, . , ], < , <= , > , >= , = , <> , ; , END, THEN, DO, TO, DOWNT0, ' , ) }
Term → factor {mulOp factor }	Follow(factor) = First(mulOp) = { * , / , DIV, MOD, AND } Follow(mulOp) = First(factor) = { NUM, STRING, FALSE, TRUE, IDENT, NOT, -, ( } Follow(factor) += Follow(term) = { * , / , DIV, MOD, AND, + , - , OR, . , ], < , <= , > , >= , = , <> , ; , END, THEN, DO, TO, DOWNT0, ' , ) }	Follow(factor) += Follow(term) = { ELSE, * , / , DIV, MOD, AND, + , - , OR, . , ], < , <= , > , >= , = , <> , ; , END, THEN, DO, TO, DOWNT0, ' , ) }
Factor → NUM   STRING   FALSE   TRUE   IDENT [ index ]   NOT factor   '-' factor   '(' expr ')'	Follow(index) += ] = { : , ] } Follow(expr) += ) = { ; , END, THEN, DO, TO, DOWNT0, ' , ) }	
RelOp → <   <=   >   >=   =   <>		
AddOp → +   -   OR		

MulOp → *   /   DIV   MOD   AND		
---------------------------------	--	--