FUNCyMonkie

**Start Symbol**

S = { function | identifier parameters ifStatementSpaced | identifier parameters ifStatementSameLine }

**Production Rules**

*P* = {

Function ::= identifier parameters equalSign expression | offside identifier parameters equalSign expression,

expression ::= relational | arithmetic | conditional | functional,

relational ::= term relationalOps term | relational relationalOps term | relational relationalOps relational,

arithmetic ::= nestExpression firstLevelArithmetic arithmetic | nestExpression | nestExpression2 | nestExpression3 | mathTerm,

nestExpression ::= nestExpression2 secondLevelArithmetic nestExpression | arithmetic,

nestExpression2 ::= expression exponent nestExpression3 | arithmetic,

nestExpression3 ::= openParen expression closeParen | arithmetic,

firstLevelArithmetic ::= +, -,

secondLevelArithmetic ::= \*, /,

exponent ::= ^,

conditional ::= ifStatementSpaced | ifStatementSameLine,

functional ::= identifier parameters | identifier openParen arithmetic closeParen | functional mathOp mathTerm | functional mathOp arithmetic,

openParen ::= (,

closeParen ::= ) ,

mathTerm ::= identifier | intLiteral | realLiteral,

term ::= identifier | intLiteral | realLiteral | booleanLiteral,

mathOp ::= +, -, /, \*, ^, >, <,

relationalOps ::= ==, /=, <=, >=,

equalSign ::= =,

ifStatementSpaced ::= ifTerminal ifCondition offside thenTerminal return offside elseTerminal return offside endifTerminal | offside ifTerminal ifCondition offside thenTerminal return offside elseTerminal return offside endifTerminal,

ifStatementSameLine ::= ifTerminal ifCondition thenTerminal return elseTerminal return endifTerminal | offside ifTerminal ifCondition thenTerminal return elseTerminal return endifTerminal,

ifCondition ::= relational | functional,

parameters ::= parameter | parameter parameters,

parameter ::= identifier,

return ::= arithmetic | intLiteral | realLiteral | identifier,

ifTerminal ::= if,

thenTerminal ::= then,

elseTerminal ::= else,

endifTerminal ::= endif,

}

**Non-Terminals**

N = { function, parameter, parameters, ifStatementSpaced, ifStatementSameLine, equalSign, expression, relational, arithmetic, conditional, functional, term, relationalOps, nestExpression, nestExpression2, nestExpression3, secondLevelArithmetic, mathTerm, exponent, openParen, closeParen, firstLevelArithmetic, mathOp, realationalOps, ifTerminal, ifCondition, thenTerminal, return, elseTerminal, endifTerminal }

**Terminals aka alphabet**

∑ = { offside, +, -, \*, /, ^, (, ), intLiteral, realLiteral, identifier, booleanLiteral, ==, >=, <=, /=, =, if, then, else, endif }