COMPILER CONSTRUCTION TERM PROJECT :

MODIFIED GRAMMAR AND FIRST & FOLLOW SETS

PRANJAL GUPTA 2013B4A7470P
TANAYA JHA 2013B3A7304P

BATCH NO : 82

GRAMMAR

1	<program>	⟶	<moduleDeclarations> <otherModules> <driverModule> <otherModules>
2	<moduleDeclarations>	⟶	<moduleDeclaration> <moduleDeclarations> | ε
3	<moduleDeclaration>	⟶	**DECLARE MODULE ID SEMICOL**
4	<otherModules>	⟶	<module> <otherModules> | ε
5	<driverModule>	⟶	**DEF DRIVER PROGRAM ENDDEF** <moduleDef>
6	<module>	⟶	**DEF MODULE ID ENDDEF TAKES INPUT SQBO** <input_plist> **SQBC SEMICOL** <ret> <moduleDef>
7	<ret>	⟶	**RETURNS SQBO** <output_plist> **SQBC SEMICOL** | ε
8	<input_plist>	⟶	**ID COLON** <dataType> <input_plistRec>
9	<input_plistRec>	⟶	**COMMA ID COLON** <dataType> <input_plistRec> | ε
10	<output_plist>	⟶	**ID COLON** <type> <output_plistRec>
11	<output_plistRec>	⟶	**COMMA ID COLON** <type> <output_plistRec> | ε
12	<type>	⟶	**INTEGER** | **REAL** | **BOOLEAN**
13	<dataType>	⟶	<type> | **ARRAY SQBO** <range> **SQBC** OF <type>
14	<moduleDef>	⟶	**START** <statements> **END**
15	<statements>	⟶	<statement> <statements> | ε
16	<statement>	⟶	<ioStmt> | <simpleStmt> | <declareStmt> | <conditionalStmt> | <iterativeStmt> | **SEMICOL**
17	<ioStmt>	⟶	**GET_VALUE BO ID** <whichId> **BC SEMICOL** | **PRINT BO** <print_val> **BC SEMICOL**

18 <print_val> ⟶ **ID** <whichId> | **NUM** | **RNUM** | **TRUE** | **FALSE**

19 <whichId> ⟶ **SQBO** <index> **SQBC** | ε

20 <index> ⟶ **NUM** | **ID**

21 <simpleStmt> ⟶ <assignmentStmt> | <moduleReuseStmt>

22 <assignmentStmt> ⟶ **ID** <whichId> **ASSIGNOP** <expression> **SEMICOL**

23 <moduleReuseStmt> ⟶ <optional> **USE MODULE ID WITH PARAMETERS** <idList> **SEMICOL**

24 <optional> ⟶ **SQBO** <idList> **SQBC ASSIGNOP** | ε

25 <idList> ⟶ **ID** <idListRec>

26 <idListRec> ⟶ **COMMA ID** <idListRec> | ε

27 <expression> ⟶ <arithmeticExpr> | <booleanExpr>

28 <arithmeticExpr> ⟶ <term> <arithmeticExprRec>

29 <arithmeticExprRec> ⟶ <pm> <term> <arithmeticExprRec> | ε

30 <term> ⟶ <factor> <termRec>

31 <termRec> ⟶ <md> <factor> <termRec> | ε

32 <factor> ⟶ **BO** <arithmeticExpr> **BC** | <var>

33 <var> ⟶ **ID** <whichId> | **NUM** | **RNUM**

34 <pm> ⟶ **PLUS** | **MINUS**

35 <md> ⟶ **MUL** | **DIV**

36 <booleanExpr> ⟶ <booleanSegment> <booleanExprRec>

37 <booleanExprRec> ⟶ <logicalOp> <booleanSegment> <booleanExprRec> | ε

38 <logicalOp> ⟶ **AND** | **OR**

39 <booleanSegment> ⟶ <arithmeticExpr> <relationalOp> <arithmeticExpr> | **BO** <booleanExpr> **BC**

40 <relationalOp> ⟶ **LT** | **LE** | **GT** | **GE** | **EQ** | **NE**

41 <declareStmt> ⟶ **DECLARE** <idList> **COLON** <dataType> **SEMICOL**

42 <conditionalStmt> ⟶ **SWITCH BO** <expression> **BC START** <caseStmt> <default> **END**

43 <caseStmt> ⟶ **CASE** <value> **COLON** <statements> **BREAK SEMICOL** <caseStmt>

44 <value> ⟶ **NUM** | **TRUE** | **FALSE**

45 <default> ⟶ **DEFAULT COLON** <statements> **BREAK SEMICOL** | ε

46 <iterativeStmt> ⟶ **FOR BO ID IN** <range> **BC START** <statements> **ENI** | **WHILE BO** <booleanExpr> **BC START** <statements> **END**

47 <range> ⟶ **NUM RANGEOP NUM**

ASSUMPTIONS

1. Logical operators AND and OR have the same priority and are left associative **[Rule 36, 37, 39]**
2. Conditional statement (SWITCH) can take argument as identifier, arithmetic expression that evaluates to NUM and boolean expression **[Rule 42]**
3. There can be 'Empty Statements' in the language. This statement consist simply of a semicolon and performs no action **[Rule 16]**
4. GET_VALUE statement can take ID as well as an element of the array of type INTEGER, REAL, BOOLEAN **[Rule 17]**
5. PRINT statement can be used to print boolean constants TRUE and FALSE as well **[Rule 18]**

FIRST AND FOLLOW SET

NONTERMINALS	FIRST SET	FOLLOW SET
<program>	DECLARE, DEF	$
<moduleDeclarations>	DECLARE, ε	DEF
<moduleDeclaration>	DECLARE	DEF, DECLARE
<otherModule>	DEF, ε	DEF, $
<module>	DEF	DEF, $
<driverModule>	DEF	DEF, $
<ret>	RETURNS, ε	START
<input_plist>	ID	SQBC
<input_plistRec>	COMMA, ε	SQBC
<output_plist>	ID	SQBC
<output_plistRec>	COMMA, ε	SQBC
<type>	INTEGER, REAL, BOOLEAN	SQBC, COMMA, SEMICOL
<dataType>	INTEGER, REAL, BOOLEAN, ARRAY	COMMA, SQBC, SEMICOL
<moduleDef>	START	DEF, $
<statements>	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, ε	BREAK, END
<statement>	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<ioStmt>	GET_VALUE, PRINT	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<print_val>	ID, NUM, RNUM, TRUE, FALSE	BC
<whichId>	SQBO, ε	MUL, DIV, SEMICOL, BC, LT,

		LE, GT, GE, EQ, NE, PLUS, MINUS, ASSIGNED
<index>	NUM, ID	SQBC
<simpleStmt>	ID, USE, SQBO	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<assignmentStmt>	ID	GET_VALUE, DECLARE, SWITCH, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<moduleReuseStmt>	SQBO, USE	GET_VALUE, PRINT, ID, SQBO, USE, DECLARE, SWITCH, FOR, SEMICOL, WHILE, BREAK, END
<optional>	SQBO, ε	USE
<idList>	ID	SEMICOL, SQBC, COLON
<idListRec>	COMMA, ε	SEMICOL, SQBC, COLON
<expression>	BO, ID, RNUM, NUM	SEMICOL, BC
<arithmeticExpr>	BO, ID, RNUM, NUM	SEMICOL, BC, LT, LE, GT, GE, EQ, NE
<arithmeticExprRec>	PLUS, MINUS, ε	SEMICOL, BC, LT, LE, GT, GE, EQ, NE
<term>	BO, ID, RNUM, NUM	SEMICOL, BC, LT, LE, GT, GE, EQ, NE, PLUS, MINUS
<termRec>	MUL, DIV, ε	SEMICOL, BC, LT, LE, GT, GE, EQ, NE, PLUS, MINUS
<factor>	BO, ID, RNUM, NUM	MUL, DIV, SEMICOL, BC, LT, LE, GT, GE, EQ, NE, PLUS, MINUS
<var>	ID, RNUM, NUM	MUL, DIV, SEMICOL, BC, LT, LE, GT, GE, EQ, NE, PLUS, MINUS
<pm>	PLUS, MINUS	BO, ID, RNUM, NUM
<md>	MUL, DIV	BO, ID, RNUM, NUM
<booleanExpr>	BO, ID, RNUM, NUM	SEMICOL, BC
<booleanExprRec>	AND, OR, ε	SEMICOL, BC

<logicalOp>	AND, OR	BO, ID, RNUM, NUM
<booleanSegment>	BO, ID, RNUM, NUM	AND, OR, SEMICOL, BC
<relationalOp>	LT, LE, GT, GE, EQ, NE	BO, ID, RNUM, NUM
<declareStmt>	DECLARE	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<conditionalStmt>	SWITCH	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<caseStmt>	CASE	DEFAULT, SWITCH
<value>	NUM, TRUE, FALSE	COLON
<default>	DEFAULT, ε	END
<iterativeStmt>	FOR, WHILE	GET_VALUE, DECLARE, SWITCH, FOR, ID, SEMICOL, SQBO, USE, PRINT, FOR, WHILE, BREAK, END
<range>	NUM	BC, SQBC

The FIRST set of a terminal is a singleton set containing only that terminal, as **FIRST(<terminal>) = {<terminal>}**