GROUP 56

Rishabh Garg (2014A7PS065P)
M Sharat Chandra (2014A7PS108P)

<program> -> <moduleDeclarations> <moduleDefinitions>

<moduleDeclarations> -> <moduleDeclaration> <moduleDeclarations> | ε
<moduleDeclaration> -> DECLARE MODULE ID SEMICOL

<moduleDefinitions> -> DEF <otherModules> <driverModule> <postDriver>
<otherModules> -> <module> DEF <otherModules> | ε
<driverModule> -> DRIVER PROGRAM ENDDEF <moduleDef>
<postDriver> -> <postModule> <postDriver> | ε
<postModule> -> DEF MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMICOL
<module> -> MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMICOL <ret> <moduleDef>
<ret> -> RETURNS SQBO <output_plist> SQBC SEMICOL | ε

<input_plist> ->   ID COLON <dataType> <input_plist_ex>
<input_plist_ex> -> COMMA ID COLON <dataType> <input_plist_ex> | ε

<output_plist> -> ID COLON <type> <output_plist_ex>
<output_plist_ex> -> COMMA ID COLON <type> <output_plist_ex> | ε

<dataType> -> INTEGER | REAL | BOOLEAN | ARRAY SQBO <range> SQBC OF <type>
<type> -> INTEGER | REAL | BOOLEAN

<moduleDef> -> START <statements> END

<statements> -> <statement> <statements> | ε
<statement> -> <ioStmt> | <simpleStmt> | <declareStmt> | <condionalStmt> | <iterativeStmt>

<ioStmt> -> GET_VALUE BO ID BC SEMICOL | PRINT BO <var> BC SEMICOL
<var> -> ID <whichId> | NUM | RNUM | TRUE | FALSE
<whichId> -> SQBO ID SQBC | ε

<simpleStmt> -> <assignmentStmt> | <moduleReuseStmt>

<assignmentStmt> -> ID <whichStmt>
<whichStmt> -> <lvalueIDStmt> | <lvalueARRStmt>
<lvalueIDStmt> -> ASSIGNOP <expression> SEMICOL
<lvalueARRStmt> -> SQBO <index> SQBC ASSIGNOP <expression> SEMICOL

<index> -> NUM | ID

<moduleReuseStmt> -> <optional> USE MODULE ID WITH PARAMETERS <idList> SEMICOL
<optional> -> SQBO <idList> SQBC ASSIGNOP | ε
<idList> -> ID<idList_ex>
<idList_ex> -> COMMA ID <idList_ex> | ε

<expression> -> <expression1> <relationalOp> <expression> | <expression1>
<expression1> -> <expression2> <logicalOp> <expression1> | <expression2>
<expression2> -> <expression3> <op1> <expression2> | <expression3>
<expression3> -> <expression4> <op2> <expression3> | <expression4>
<expression4> -> BO <expression> BC | <var>

<op1> -> PLUS | MINUS
<op2> -> MUL | DIV
<relationalOp> -> LT | LE | GT | GE | EQ | NE
<logicalOp> -> AND | OR

<declareStmt> -> DECLARE <idList> COLON <dataType> SEMICOL

<condionalStmt> -> SWITCH BO ID BC START <caseStmt> <default> END
<caseStmt> -> CASE <value> COLON <statements> BREAK SEMICOL <caseStmt>
<value> -> NUM | TRUE | FALSE
<default> -> DEFAULT COLON <statements> BREAK SEMICOL | ε

<iterativeStmt> -> FOR BO ID IN <range> BC START <statements> END | WHILE BO <expression> BC START <statements> END
<range> -> NUM RANGEOP NUM

NOTES about syntax analyser:

1. It cannot determine if a token is declared before it is being used
2. It cannot determine if a token is initialized before it is being used
3. It cannot determine if an operation performed on a token type is valid or not

Assumptions:

There is no distinction between Arithmetic and Boolean Expressions. All of them are parsed using standard precedence of operators. We do not check for validity of expressions here. It will be done in the semantic analyser.