## Group 5 - AST Rules

Patel Darsh Rajesh - 2018B4A70532P
Purusharth Amrut - 2018B5A70897P
Harsh Jhunjhunwala - 2018B5A70691P
Kush Mehta - 2018B5A70956P
Jatin Aggarwal - 2018B4A70884P

GRAMMAR RULES

1. <program> ===> <otherFunctions> <mainFunction>
2. <mainFunction> ===> TK_MAIN <stmts> TK_END
3. <otherFunctions> ===> <function> <otherFunctions>
3. <otherFunctions> ===> eps
4. <function> ===> TK_FUNID <input_par> <output_par> TK_SEM <stmts>TK_END
5. <input_par> ===> TK_INPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR
6. <output_par> ===> TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR
6. <output_par> ===> eps
7. <parameter_list> ===> <dataType> TK_ID <remaining_list>
8. <dataType> ===> <primitiveDatatype>
8. <dataType> ===> <constructedDatatype>
9. <primitiveDatatype> ===> TK_INT
9. <primitiveDatatype> ===> TK_REAL
10. <constructedDatatype> ===> TK_RECORD TK_RUID
10. <constructedDatatype>===> TK_UNION TK_RUID
10. <constructedDatatype>===> TK_RUID
11. <remaining_list> ===> TK_COMMA <parameter_list>
11. <remaining_list> ===> eps

12. <stmts> ===> <typeDefinitions> <declarations> <otherStmts> <returnStmt>
13. <typeDefinitions> ===> <actualOrRedefined> <typeDefinitions>
13. <typeDefinitions> ===> eps
14. <actualOrRedefined> ===> <typeDefinition>
14. <actualOrRedefined> ===> <definetypestmt>
15. <typeDefinition>===>TK_RECORD TK_RUID <fieldDefinitions> TK_ENDRECORD
15. <typeDefinition>===>TK_UNION TK_RUID <fieldDefinitions> TK_ENDUNION
16. <fieldDefinitions> ===> <fieldDefinition1> <fieldDefinition2> <moreFields>
17. <fieldDefinition>===> TK_TYPE <fieldType > TK_COLON TK_FIELDID TK_SEM
18. <fieldType>===> <primitiveDatatype>
18. <fieldtype>===> TK_RUID
19. <moreFields> ===> <fieldDefinition><moreFields>
19. <moreFields> ===> eps
20. <declarations> ===> <declaration><declarations1>
20. <declarations> ===> eps
21. <declaration>===> TK_TYPE <dataType> TK_COLON TK_ID <global_or_not> TK_SEM
22. <global_or_not> ===> TK_COLON TK_GLOBAL
22. <global_or_not> ===> eps
23. <otherStmts> ===> <stmt> <otherStmts1>
23. <otherStmts> ===> eps
24. <stmt> ===> <assignmentStmt>
24. <stmt> ===> <iterativeStmt>
24. <stmt> ===> <conditionalStmt>
24. <stmt> ===> <ioStmt>

24. <stmt> ===> <funCallStmt>
25. <assignmentStmt> ===> <singleOrRecId> TK_ASSIGNOP <arithmeticExpression> TK_SEM
26. <singleOrRecId> ===> TK_ID<option_single_constructed>
27. <option_single_constructed> ===> eps
27. <option_single_constructed> ===> <oneExpansion><moreExpansions>
28. <oneExpansion>===> TK_DOT TK_FIELDID
29. <moreExpansions>===> <oneExpansion> <moreExpansions1> |
29. <moreExpansions>===> eps
30. <funCallStmt> ===> <outputParameters> TK_CALL TK_FUNID TK_WITH TK_PARAMETERS <inputParameters> TK_SEM
31. <outputParameters> ===> TK_SQL <idList> TK_SQR TK_ASSIGNOP
31. <outputParameters> ===> eps
32. <inputParameters> ===> TK_SQL <idList> TK_SQR
33. <iterativeStmt> ===> TK_WHILE TK_OP <booleanExpression> TK_CL <stmt> <otherStmts> TK_ENDWHILE
34. <conditionalStmt> ===> TK_IF TK_OP <booleanExpression> TK_CL TK_THEN <stmt> <otherStmts> <elsePart>
35. <elsePart> ===> TK_ELSE <stmt> <otherStmts> TK_ENDIF
35. <elsePart> ===> TK_ENDIF
36. <ioStmt> ===> TK_READ TK_OP <var> TK_CL TK_SEM
36. <ioStmt> ===> TK_WRITE TK_OP <var> TK_CL TK_SEM
37. <arithmeticExpression> ===> <term> <expPrime>
38. <expPrime> ===> <lowPrecedenceOperators> <term> <expPrime1>
38. <expPrime> ===> eps
39. <term>===> <factor> <termPrime>
40. <termPrime> ===> <highPrecedenceOperators><factor> <termPrime>
40. <termPrime> ===> eps
41. <factor> ===> TK_OP <arithmeticExpression> TK_CL

41. <factor> ===> <var>
42. <highPrecedenceOperator>===> TK_MUL
42. <highPrecedenceOperator>===> TK_DIV
43. <lowPrecedenceOperators> ===> TK_PLUS
43. <lowPrecedenceOperators> ===> TK_MINUS
44. <booleanExpression> ===> TK_OP <booleanExpression1> TK_CL <logicalOp> TK_OP <booleanExpression2> TK_CL
44. <booleanExpression> ===> <var> <relationalOp> <var>
44. <booleanExpression> ===> TK_NOT TK_OP <booleanExpression> TK_CL
45. <var>===> <singleOrRecId
45. <var>===> TK_NUM
45. <var>===> TK_RNUM
46. <logicalOp> ===> TK_AND
46. <logicalOp> ===> TK_OR
47. <relationalOp> ===> TK_LT
47. <relationalOp> ===> TK_LE
47. <relationalOp> ===> TK_EQ
47. <relationalOp> ===> TK_GT
47. <relationalOp> ===> TK_GE
47. <relationalOp> ===> TK_NE
48. <returnStmt> ===> TK_RETURN <optionalReturn> TK_SEM
49. <optionalReturn> ===> TK_SQL <idList> TK_SQR
49. <optionalReturn> ===> eps
50. <idList> ===> TK_ID <more_ids>
51. <more_ids> ===> TK_COMMA <idList>
51. <more_ids> ===> eps

# AST RULES

1. program.ptr = mkNode(program, otherFunctions.ptr, mainFunction.ptr)
2. mainFunction.ptr = stmts.ptr
3. otherFunctions.ptr = mkNode(otherFunctions, function.ptr, otherFunctions.ptr)
3. otherFunctions.ptr = NULL
4. function.ptr = mkNode(function, mkLeaf(funtionId, entry.TK_FUNID), input_par.ptr, output_par.ptr, stmts.ptr)
5. input_par.ptr = parameter_list.ptr
6. output_par.ptr = parameter_list.ptr
6. output_par.ptr = NULL
7. parameter_list.ptr = mkNode(parameter_list, dataType.ptr, mkLeaf(id, entry.TK_ID), remaining_list.ptr)
8. dataType.ptr = constructedDatatype.ptr
8. dataType.ptr = primitiveDatatype.ptr
9. primitiveDatatype.ptr = mkLeaf(int, null)
9. primitiveDatatype.ptr = mkLeaf(real, null)
10. constructedDatatype.ptr = mkleaf(record, entry.TK_RUID)
10. constructedDatatype.ptr = mkleaf(union, entry.TK_RUID)
10. constructedDatatype.ptr = mkLeaf(recordOrUnion, entry.TK_RUID)
11. remaining_list.ptr = parameter_list.ptr
11. remaining_list.ptr = NULL
12. stmts.ptr = mkNode(stmts, typeDefinitions.ptr, declarations.ptr, otherStmts.ptr, returnStmt.ptr)
13. typeDefinitions.ptr = mkNode(typeDefinitions ,actualOrRedefined.ptr, typeDefinitions.ptr)
13. typeDefinitions.ptr = NULL
14. actualOrRedefined.ptr = typeDefinition.ptr
14. actualOrRedefined.ptr = definetypestmt.ptr
15. typeDefinition.ptr = mkNode(typeDefinition, mkLeaf(record, entry.TK_RUID), fieldDefinitions.ptr)
15. typeDefinition.ptr = mkNode(typeDefinition, mkLeaf(union, entry.TK_RUID), fieldDefinitions.ptr)
16.fieldDefinitions.ptr = mkNode(fieldDefinitions, fieldDefinition1.ptr, fieldDefinition2.ptr, moreFields.ptr)
17. fieldDefinition.ptr = mkNode(fieldDefinition, fieldType.ptr, mkLeaf(fieldId, entry.TK_FIELDID))
18. fieldType.ptr = primitiveDatatype.ptr
18. fieldType.ptr = mkLeaf(recordOrUnion, entry.TK_RUID)

19.moreFields.ptr = mkNode(moreFields, fieldDefinition.ptr, moreFields.ptr)
19.moreFields.ptr = NULL
20.declarations.ptr = mkNode(declarations, declaration.ptr, declarations1.ptr)
20.declarations.ptr = NULL
21.declaration.ptr = mkNode(declaration, dataType.ptr, mkLeaf(id, entry.TK_ID, global_or_not.ptr)
22.global_or_not.ptr = mkLeaf(global, null)
22.global_or_not.ptr = NULL
23.otherStmts.ptr = mkNode(otherStmts, stmt.ptr, otherStmts1.ptr)
23.otherStmts.ptr = NULL
24.stmt.ptr = assignmentStmt.ptr
24.stmt.ptr = iterativeStmt.ptr
24.stmt.ptr = conditionalStmt.ptr
24.stmt.ptr = ioStmt.ptr
24.stmt.ptr = funCallStmt.ptr
25.assignmentStmt.ptr = mkNode(assignmentStmt, singleOrRecId.ptr, arithmeticExpression.ptr)
26. singleOrRecId.ptr = mkNode(singleOrRecId, mkLeaf(id, entry.TK_ID), option_single_constructed.ptr)
27. option_single_constructed.ptr = NULL
27. option_single_constructed.ptr = mkNode(option_single_constructed, oneExpansion.ptr, moreExpansions.ptr)
28. oneExpansion.ptr = mkLeaf(fieldId, entry.TK_FIELDID)
29. moreExpansions.ptr = mkNode(moreExpansions, oneExpansion.ptr, moreExpansions1.ptr)
29. moreExpansions.ptr = NULL
30. funCallStmt.ptr = mkNode(funCallStmt, ouputParameters.ptr, mkLeaf(funId, entry.TK_FUNID), inputParameters.ptr)
31.outputParameters.ptr = idList.ptr
31.outputParameters.ptr = NULL
32.inputParameters.ptr = idList.ptr
33.iterativeStmt.ptr = mkNode(iterativeStmt, booleanExpression.ptr, stmt.ptr, otherStmts.ptr)
34.conditionalStmt.ptr = mkNode(conditionalStmt, booleanExpression.ptr, stmt.ptr, otherStmts.ptr, elsePart.ptr)
35. elsePart.ptr = mkNode(elsePart, stmt.ptr, otherStmts.ptr)
36. ioStmt.ptr = var.ptr
36. ioStmt.ptr = var.ptr
37.arithmeticExpression.ptr = mkNode(arithmeticExpression, term.ptr, expPrime.ptr)
38.expPrime.ptr = mkNode(expPrime, lowPrecedenceOperator.ptr, term.ptr, expPrime1.ptr)
38.expPrime.ptr = NULL

39. term.ptr = mkNode(term, factor.ptr, termPrime.ptr)
40. termPrime.ptr = mkNode(termPrime, highPrecedenceOperator.ptr, factor.ptr, termPrime.ptr)
40. termPrime.ptr = NULL
41. factor.ptr = arithmeticExpression.ptr
41. factor.ptr = var.ptr
42. highPrecedenceOperator.ptr = mkLeaf(mul, null)
42. highPrecedenceOperator.ptr = mkLeaf(div, null)
43. lowPrecedenceOperators.ptr = mkLeaf(plus, null)
43. lowPrecedenceOperators.ptr = mkLeaf(minus, null)
44. booleanExpression.ptr = mkNode(booleanExpression, booleanExpression1.ptr, logicalOp.ptr, booleanExpression2.ptr)
44.booleanExpression.ptr = mkNode(booleanExpression, var.ptr, relationalOp.ptr, var.ptr)
44. booleanExpression.ptr = mkNode(booleanExpression, TK_NOT.ptr, booleanExpression.ptr)
45. var.ptr = singleOrRecID.ptr
45. var.ptr = mkLeaf(num, null)
45. var.ptr = mkLeaf(rnum, null)
46. logicalOp.ptr = mkLeaf(and, null)
46. logicalOp.ptr = mkLeaf(or, null)
47. relationalOp.ptr = mkLeaf(lt, null)
47. relationalOp.ptr = mkLeaf(le, null)
47. relationalOp.ptr = mkLeaf(eq, null)
47. relationalOp.ptr = mkLeaf(gt, null)
47. relationalOp.ptr = mkLeaf(ge, null)
47. relationalOp.ptr = mkLeaf(ne, null)
48. returnStmt.ptr = optionalReturn.ptr
49. optionalReturn.ptr = idList.ptr
49. optionalReturn.ptr = NULL
50. idList.ptr = mkNode(idList, mkLeaf(id, entry.TK_ID), more_ids.ptr)
51. more_ids.ptr = idList.ptr
51. more_ids.ptr = NULL