AST RULES

GROUP – 14

VARUN GUPTA (2016A7PS0087P)
TARUN KUMAR(2016A7PS0005P)
AVICHAL JAIN(2016A7PS0046P)
VAIBHAV MAHESHWARI(2016A7PS0081P)

```
<program> ===> <otherFunctions> <mainFunction>
        program.node= newNode(ROOT_NODE, concatList(otherFunctions.children, mainFunction.node))
        free(otherFunctions)
<mainFunction>===> TK_MAIN <stmts> TK_END
        mainFunction.node = stmts.node
<otherFunctions> ===> <function> <otherFunctions1>
        otherFunctions.node = concatList(function.node, otherFunctions1.children)
        free(otherFunctions1)
<otherFunctions> ===>eps
        otherFunctions.node = NULL
<function> ===> TK_FUNID <input_par> <output_par> TK_SEM  <stmts>  TK_END
        function.node = newNode(FUNC_NODE, concatList(FUNID.node, input_par.node, output_par.node,   stmts.node))
<input_par> ===> TK_INPUT  TK_PARAMETER  TK_LIST TK_SQL  <parameter_list>  TK_SQR
        input_par.children = parameter_list.children
        free(parameter_list)
<output_par> ===> TK_OUTPUT  TK_PARAMETER  TK_LIST TK_SQL  <parameter_list> TK_SQR
        output_par.children = parameter_list.children
        free(parameter_list)
<output_par> ===> eps
        output_par.children = NULL
<parameter_list> ===> <dataType>  TK_ID  <remaining_list>
        parameter_list.children = newNode(PAR_NODE, concatList(dataType.node, ID.node,remaining_list.children))
        free(remaining_list)
        free(datatype)
<dataType> ===>  <primitiveDatatype>
        dataType.node = primitiveDatatype.node
        free(primitiveDatatype)
<dataType> ===> <constructedDatatype>
        dataType.node = constructedDatatype.node
        free(constructedDatatype)
<primitiveDatatype> ===> TK_INT
        primitiveDatatype.node=makeLeaf(INT_NODE, INT.node)
<primitiveDatatype> ===> TK_REAL
        primitiveDatatype.node=makeLeaf(REAL_NODE, REAL.node)
<constructedDatatype> ===> TK_RECORD  TK_RECORDID
        constructedDatatype.node = makeLeaf(RECORDID_NODE, RECORDID.node)
<remaining_list> ===> TK_COMMA <parameter_list>
        remaining_list.children=parameter_list.children
<remaining_list> ===> eps
        remaining_list.children=NULL
<stmts> ===> <typeDefinitions> <declarations> <otherStmts> <returnStmt>
        stmts.node = concatList(typeDefinitions.node, declarations.node, otherStmts.children, returnStmt.node)
        free(otherStmts)
<typeDefinitions> ===> <typeDefinition> <typeDefinitions1>
        typeDefinitions.children = concatList(typeDefinition.node, typeDefinitions1.children)
        free(typeDefinitions1)
<typeDefinitions> ===>  eps
        typeDefinitions.children=NULL
<typeDefinition> ===> TK_RECORD TK_RECORDID <fieldDefinitions> TK_ENDRECORD TK_SEM
        typeDefinition.children = newNode(TYPE_DEFINITION_NODE, concatList(RECORDID.node, fieldDefinitions.children))
        free(fieldDefinitions)
<fieldDefinitions> ===> <fieldDefinition1> <fieldDefinition2> <moreFields>
        fieldDefinitions.children = concatList(fieldDefinition1.node, fieldDefinition2.node, moreFields.children)
        free(morefields)
<fieldDefinition> ===> TK_TYPE <primitiveDatatype> TK_COLON TK_FIELDID TK_SEM
        fieldDefinition.children = newNode(FIELD_DEFINITION_NODE, concatList(primitiveDatatype.node, FIELDID.node))
        free(primitiveDatatype)
<moreFields> ===> <fieldDefinition> <moreFields1>
        moreFields.children = concatList(fieldDefinition.Node,moreFields1.children)
        free(morefields1)
<moreFields> ===> eps
        morefields.children=NULL
<declarations> ===> <declaration> <declarations1>
        declarations.children= concatList(declaration.node,  declarations1.children)
        free( declarations1)
<declarations> ===> eps
        declarations.children=NULL
```

```
<declaration> ===> TK_TYPE <dataType> TK_COLON TK_ID  <global_or_not> TK_SEM
        declaration.children = newNode(DECLARATION_NODE, concatList( dataType.node, ID.node, global_or_not.node))
        free(datatype)
        free(global_or_not)
<global_or_not> ===> TK_COLON  TK_GLOBAL
        global_or_not.node=GLOBAL.node
<global_or_not> ===> eps
        global_or_not.node=NULL
<otherStmts> ===> <stmt> <otherStmts1>
        otherStmts.children = concatList(stmt.node, otherstmts1.children)
        free(stmt)
        free(otherstmts1)
<otherStmts> ===> eps
        otherStmts.children=NULL
<stmt> ===> <assignmentStmt>
        stmt.node= assignmentStmt.node
<stmt> ===> <funCallStmt>
        stmt.node= funCallStmt.node
<stmt> ===> <iterativeStmt>
        stmt.node= iterativeStmt.node
<stmt> ===> <conditionalStmt>
        stmt.node= conditionalStmt.node
<stmt> ===> <ioStmt>
        stmt.node= ioStmt.node
<assignmentStmt> ===> <singleOrRecId> TK_ASSIGNOP  <arithmeticExpression> TK_SEM
        assignmentStmt.children = concatList( singleOrRecId.node, arithmeticExpression.node)
<singleOrRecId> ===> TK_ID  <new_24>
        singleOrRecId.children = newNode(ID_NODE, concatList(ID.node, new_24.node))
        free(new_24)
<new_24> ===> eps
        new_24.node= NULL
<new_24> TK_DOT TK_FIELDID
        new_24.node = FIELDID.node
<funCallStmt> ===> <outputParameters> TK_CALL  TK_FUNID  TK_WITH TK_PARAMETERS <inputParameters>  TK_SEM
        funCallStmt.children = newNode(FUNC_CALL_NODE, concatList( outputParameters.node, FUNID.node, inputParameters.node))
<outputParameters> ==> TK_SQL <idList> TK_SQR TK_ASSIGNOP
        outputParameters.children = concatList( idList.children)
        free(idList)
<outputParameters> ==> eps
        outputParameters.children=NULL
<inputParameters> ===> TK_SQL <idList> TK_SQR
        inputParameters.children = idList.children
        free(idList)
<iterativeStmt> ===> TK_WHILE TK_OP <booleanExpression> TK_CL <stmt> <otherStmts> TK_ENDWHILE
        iterativeStmt.children = concatList( booleanExpression.node,stmt.node, otherStmts.children)
        free(stmt)
        free(booleanexpression)
        free(otherStmts)
<conditionalStmt> ===> TK_IF TK_OP <booleanExpression> TK_CL TK_THEN <stmt> <otherStmts> <elsePart>
        conditionalstmt.children = concatList( booleanExpression.node,stmt.node, otherStmts.children, elsePart.node)
        free(stmt)
        free(booleanexpression)
        free(otherStmts)
<elsePart> ===> TK_ELSE <stmt> <otherStmts> TK_ENDIF
        elsePart.children = concatList(stmt.node, otherstmts.children)
        free(stmt)
        free(otherStmts)
<elsePart> ===> TK_ENDIF
        elsePart.children=NULL
<ioStmt> ===> TK_READ TK_OP <singleOrRecId> TK_CL TK_SEM
        ioStmt.children = concatList(read.node, singleOrRecId.node)
<ioStmt> ===>  TK_WRITE TK_OP <allVar> TK_CL TK_SEM
        ioStmt.children = concatList(write.node, allvar.node)
<allVar> ===> TK_NUM
        allVar.node = makeLeaf(NUM_NODE, NUM.node)
<allVar> ===> TK_RNUM
        allVar.node = makeLeaf(RNUM_NODE, RNUM.node)
<allVar> ===> TK_ID <tempvar>
        allVar.node = newNode(ALLVAR_NODE, concatlList(ID.node, tempvar.node))
<tempvar> ===> TK_DOT TK_FIELDID
        tempvar.node = makeLeaf(FIELDID_NODE, FIELDID.node)
<tempvar> ===> eps
        tempvar.node=NULL
<arithmeticExpression> ===> <term> <expPrime>
        arithmeticExpression.node = concatList(term.node, expPrime.children)
        free(expPrime)
<expPrime> ===> <lowPrecedenceOperators> <term> <expPrime1>
        expPrime.children = newNode(EXPPRIME_NODE, concatList(lowPrecedenceOperators.node, term.node, expPrime1.children))
```

```
<expPrime> ===> eps
        expPrime.children = NULL
<term> ===> <factor> <termPrime>
        term.node = concatList(factor.node, termPrime.children)
        free(termPrime)
<termPrime> ===> <highPrecedenceOperators> <factor> <termPrime1>
        termPrime.children  = newNode(TERMPRIME_NODE, concatList(highPrecedenceOperators.node, factor.node, termPrime1.children))
<termPrime> ===> eps
        termPrime.children=NULL
<factor> ===> TK_OP <arithmeticExpression> TK_CL
        factor.node = arithmeticExpression.node
<factor> ===> <all>
        factor.node = all.node
<highPrecedenceOperators> ===> TK_MUL
        highPrecedenceOperators.node = makeLeaf(MUL_NODE, MUL.node)
<highPrecedenceOperators> ===> TK_DIV
        highPrecedenceOperators.node =makeLeaf(DIV_NODE, DIV.node)
<lowPrecedenceOperators> ===> TK_PLUS
        lowPrecedenceOperators.node = makeLeaf(PLUS_NODE, PLUS.node)
<lowPrecedenceOperators> ===> TK_MINUS
        lowPrecedenceOperators.node = makeLeaf(MINUS_NODE, MINUS.node)
<all> ===> TK_NUM
        all.node = makeLeaf(NUM_NODE, NUM.node)
<all> ===> TK_RNUM
        all.node = makeLeaf(RNUM_NODE, RNUM.node)
<all> ===> TK_ID temp
        all.node = newNode(ALL_NODE, concatList(ID.node, temp.node))
<temp> ===> TK_DOT TK_FIELDID
        temp.node = makeLeaf(FIELDID_NODE, FIELDID.node)
<temp> ===> eps
         temp.node=NULL
 <booleanExpression>===>TK_OP <booleanExpression> TK_CL <logicalOp> TK_OP<booleanExpression> TK_CL
        booleanExpression.node= logicalOp.node
        logicalOp.children = concatList( booleanExpression1.node, booleanExpression2.node)
        free(logicalOp)
        free(booleanExpression1)
        free(booleanExpression2)
<booleanExpression>===> <var> <relationalOp> <var>
        booleanExpression.node= relationalOp.node
        relationalOp.children = concatList(var1.node, var2.node)
        free(var1)
        free(var2)
        free(relationalOp)
<booleanExpression>===> TK_NOT <booleanExpression>
        booleanExpression.node=NOT.node
        NOT.children= (booleanExpression1.node)
        free(booleanExpression1)
<var>===> TK_ID
        var.node= makeLeaf(ID_NODE, ID.node)
<var>===> TK_NUM
        var.node= makeLeaf(NUM_NODE, NUM.node)
<var>===> TK_RNUM
        var.node= makeLeaf(RNUM_NODE, RNUM.node)
<logicalOp>===>TK_AND
        logicalOp.node= makeLeaf(AND_NODE, AND.node)
<logicalOp>===>TK_OR
        logicalOp.node= makeLeaf(OR_NODE, OR.node)
<relationalOp>===> TK_LT
        relationalOp.node=makeLeaf(LT_NODE, LT.node)
<relationalOp>===> TK_LE
        relationalOp.node=makeLeaf(LE_NODE, LE.node)
<relationalOp>===> TK_EQ
        relationalOp.node=makeLeaf(EQ_NODE, EQ.node)
<relationalOp>===> TK_GT
        relationalOp.node=makeLeaf(GT_NODE, GT.node)
<relationalOp>===> TK_GE
        relationalOp.node=makeLeaf(GE_NODE, GE.node)
<relationalOp>===> TK_NE
        relationalOp.node=makeLeaf(NE_NODE, NE.node)
<returnStmt>===>TK_RETURN <optionalReturn> TK_SEM
        returnStmt.children = optionalReturn.children
        free(optionalReturn)
<optionalReturn>===>TK_SQL <idList> TK_SQR
        optionalReturn.children=idList.children
        free(idList)
<optionalReturn>===> eps
        optionalReturn.children=NULL
<idList>===> TK_ID <more_ids>
```

```
        idList.children = concatList(ID.node, more_ids.children)
        free(more_ids)

<more_ids>===> TK_COMMA <idList>

        more_ids.children=idList.children

<more_ids>===> eps

        more_ids.children=NULL
```