# Modified Grammar Group 6

1. **<program>** ===> <otherFunctions> <mainFunction>

   FIRST: {TK_MAIN, TK_FUNID}

   FOLLOW: {$}

2. <mainFunction>===> TK_MAIN <stmts> TK_END

   FIRST: {TK_MAIN}

   FOLLOW: {$}

3. <otherFunctions>===> <function><otherFunctions> | ∈

   FIRST: {TK_FUNID, ∈}

   FOLLOW: {TK_MAIN}

4. <function>===>TK_FUNID <input_par> <output_par> TK_SEM <stmts> TK_END

   FIRST: {TK_FUNID}

   FOLLOW: {TK_FUNID, TK_MAIN}

5. <input_par>===>TK_INPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR

   FIRST: {TK_INPUT}

   FOLLOW: {TK_OUTPUT, TK_SEM}

6. <output_par>===>TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR| ∈

   FIRST: {TK_OUTPUT, ∈}

   FOLLOW: {TK_SEM}

7. <parameter_list>===><dataType> TK_ID <remaining_list>

   FIRST: {TK_INT, TK_REAL, TK_RECORD, TK_UNION, TK_RUID}

   FOLLOW: {TK_SQR, }

8. <dataType>===> <primitiveDatatype> |<constructedDatatype>

   FIRST: {TK_INT, TK_REAL, TK_RECORD, TK_UNION, TK_RUID}

   FOLLOW: {TK_ID, TK_COLON}

9. <primitiveDatatype>===> TK_INT | TK_REAL

   FIRST: {TK_INT, TK_REAL}

   FOLLOW: {TK_ID, TK_COLON}

10. <constructedDatatype>===>TK_RECORD TK_RUID | TK_UNION TK_RUID | TK_RUID

    FIRST: {TK_RECORD, TK_UNION, TK_RUID}

    FOLLOW: {TK_ID, TK_COLON}

    FIRST(TK_RECORD) = {TK_RECORD}, FIRST(TK_UNION) = {TK_UNION}, FIRST(TK_RUID) = {TK_RUID}.

    This shows that FIRST(TK_RECORD), FIRST(TK_UNION), FIRST(TK_RUID) are all disjoint.

This implies that this rule is in LL(1).

11. <remaining_list>===>TK_COMMA <parameter_list> | ∈

FIRST: {TK_COMMA, ∈}

FOLLOW: {TK_SQR}

12. <stmts>===><typeDefinitions> <declarations> <otherStmts><returnStmt>

FIRST: {TK_TYPE}

FOLLOW: {TK_END}

13. <typeDefinitions>===><typeDefinition><typeDefinitions> | <definetypestmt><typeDefinitions> | ∈

FIRST: {TK_RECORD, TK_UNION, TK_DEFINETYPE}

FOLLOW: {TK_TYPE}

FIRST(<typeDefinition>) = {TK_RECORD, TK_UNION},
FIRST(<definetypestmt>) = {TK_DEFINETYPE}, FIRST(∈) = {∈},
FOLLOW(<typeDefinitions>) = {TK_TYPE}.

This shows that FIRST(<typeDefinition>), FIRST(<definetypestmt>), FIRST(∈) are all disjoint, FOLLOW(<typeDefinitions>) ∩ FIRST(<typeDefinition>) = ϕ and FOLLOW(<typeDefinitions>) ∩ FIRST(<typeDefinition>) = ϕ.

This implies that this rule is in LL(1).

14. <typeDefinition>===>TK_RECORD TK_RUID <fieldDefinitions> TK_ENDRECORD

FIRST: {TK_RECORD, TK_UNION}

FOLLOW: {TK_RECORD, TK_UNION, TK_DEFINETYPE, TK_TYPE}

15. <typeDefinition>===>TK_UNION TK_RUID <fieldDefinitions> TK_ENDUNION

    FIRST: {TK_RECORD, TK_UNION}

    FOLLOW: {TK_RECORD, TK_UNION, TK_DEFINETYPE, TK_TYPE}

16. <fieldDefinitions>===> <fieldDefinition><fieldDefinition><moreFields>

    FIRST: {TK_TYPE, TK_RECORD, TK_UNION}

    FOLLOW: {TK_ENDRECORD, TK_ENDUNION}

17. <fieldDefinition>===> TK_TYPE <dataType>TK_COLON TK_FIELDID TK_SEM | <typeDefinition><fieldDefinition>

    FIRST: {TK_TYPE, TK_RECORD, TK_UNION}

    FOLLOW: {TK_TYPE, TK_RECORD, TK_UNION, TK_ENDRECORD, TK_ENDUNION}

    FIRST(TK_TYPE) = {TK_TYPE}, FIRST(<typeDefinition>) = {TK_RECORD, TK_UNION}.

    This shows that FIRST(<typeDefinition>) and FIRST(TK_TYPE) are disjoint.

    This implies that this rule is in LL(1).

18. <moreFields>===><fieldDefinition><moreFields> | ∈

    FIRST: {TK_TYPE, TK_RECORD, TK_UNION, ∈}

    FOLLOW: { TK_ENDRECORD, TK_ENDUNION}

19. <declarations> ===> <declaration><declarations> | ∈

    FIRST: {TK_TYPE, ∈}

FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_RETURN, TK_CALL}

20. <declaration>===> TK_TYPE <dataType> TK_COLON TK_ID <global_or_not> TK_SEM

    FIRST: {TK_TYPE}

    FOLLOW: {TK_TYPE, TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_RETURN, TK_CALL}

    FIRST(TK_TYPE) = {TK_TYPE}.

    As there is just one production this rule is already in LL(1).

21. <global_or_not>===>TK_COLON TK_GLOBAL| ∈

    FIRST: {TK_COLON, ∈}

    FOLLOW: {TK_SEM}

    FIRST(TK_COLON) = {TK_COLON}, FIRST(∈) = {∈}, FOLLOW(<global_or_not>) = {TK_SEM}.

    This shows that FIRST(<TK_COLON>) and FIRST(∈) are disjoint, and FOLLOW(<global_or_not>) ∩ FIRST(TK_COLON) = ϕ.

    This implies that this rule is in LL(1).

22. <otherStmts>===> <stmt><otherStmts> | ∈

    FIRST: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, ∈}

    FOLLOW: {TK_RETURN, TK_ENDIF, TK_ELSE, TK_ENDWHILE}

23. <stmt>===> <assignmentStmt> | <iterativeStmt>|<conditionalStmt>|<ioStmt>| <funCallStmt>

FIRST: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, ∈}

FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF, TK_RETURN, TK_CALL }

24. <assignmentStmt>===><singleOrRecId> TK_ASSIGNOP <arithmeticExpression> TK_SEM

FIRST: {TK_ID}

FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF, TK_RETURN, TK_CALL }

25. <singleOrRecId>===>TK_ID <nestedDot>

FIRST: {TK_ID}

FOLLOW: {TK_ASSIGNOP, TK_CL, TK_MUL, TK_DIV, TK_ID, TK_NUM, TK_RNUM, TK_OP}

FIRST(TK_ID) = {TK_ID}.

As there is just one production this rule is already in LL(1).

26. <nestedDot> ===> TK_DOT TK_FIELDID | ∈

FIRST: {TK_DOT, ∈}

FOLLOW: {TK_ASSIGNOP, TK_CL, TK_MUL, TK_DIV, TK_ID, TK_NUM, TK_RNUM, TK_OP}

FIRST(TK_DOT) = {TK_DOT}, FIRST(∈) = {∈}, FOLLOW(<nestedDot>) = {TK_ASSIGNOP, TK_CL, TK_MUL, TK_DIV, TK_ID, TK_NUM, TK_RNUM, TK_OP}.

This shows that FIRST(TK_DOT) and FIRST(∈) are disjoint, and FOLLOW(<nestedDot>) ∩ FIRST(TK_DOT) = ϕ.

This implies that this rule is in LL(1).

27. <funCallStmt>===><outputParameters> TK_CALL TK_FUNID TK_WITH TK_PARAMETERS <inputParameters> TK_SEM

FIRST: {TK_SQL, TK_CALL}

FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF, TK_RETURN, TK_CALL },

FIRST(<outputParameters>) = {TK_SQL, ∈}

As there is just one production this rule is already in LL(1).

28. <outputParameters> ==> TK_SQL <idList> TK_SQR TK_ASSIGNOP | ∈

FIRST: {TK_SQL, ∈}

FOLLOW: {TK_CALL}

29. <inputParameters>===> TK_SQL <idList> TK_SQR

FIRST: {TK_SQL}

FOLLOW: {TK_SEM}

30. <iterativeStmt>===> TK_WHILE TK_OP <booleanExpression> TK_CL <stmt><otherStmts> TK_ENDWHILE

FIRST: {TK_WHILE}

FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF, TK_RETURN, TK_CALL }

31. <conditionalStmt>===> TK_IF TK_OP <booleanExpression> TK_CL TK_THEN <stmt><otherStmts> <optionalElse> TK_ENDIF

FIRST: {TK_IF}

FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF, TK_RETURN, TK_CALL }

FIRST(TK_IF) = {TK_IF}.

As there is just one production this rule is already in LL(1).

32. <optionalElse>===> TK_ELSE <stmt><otherStmts> | ∈

    FIRST: {TK_ELSE, ∈}

    FOLLOW: {TK_ENDIF}

    FIRST(TK_ELSE) = {TK_MUL, TK_DIV}, FIRST(∈) = {∈}, FOLLOW(<optionalElse>) = {TK_ENDIF}.

    This shows that FIRST(TK_ELSE) and FIRST(∈) are disjoint, and FOLLOW(<optionalElse>) ∩ FIRST(TK_ELSE) = ϕ.

    This implies that this rule is in LL(1).

33. <ioStmt>===>TK_READ TK_OP <var> TK_CL TK_SEM | TK_WRITE TK_OP <var> TK_CL TK_SEM

    FIRST: {TK_READ, TK_WRITE}

    FOLLOW: {TK_ID, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF, TK_RETURN, TK_CALL }

34. <arithmeticExpression> ====> <term><N1>

    FIRST: {TK_ID, TK_NUM, TK_RNUM, TK_OP}

    FOLLOW: {TK_CL, TK_SEM}

    As there is just one production this rule is already in LL(1).

35. <N1>===><operator1><term><N1> | ∈

    FIRST: {TK_PLUS, TK_MINUS, ∈}

    FOLLOW: {TK_CL, TK_SEM}

    FIRST(<operator1>) = {TK_PLUS, TK_MINUS}, FIRST(∈) = {∈}, FOLLOW(<N1>) = {TK_CL, TK_SEM}.

    This shows that FIRST(<operator1>) and FIRST(∈) are disjoint, and FOLLOW(<N1>) ∩ FIRST(<operator1>) = ϕ.

    This implies that this rule is in LL(1).

36. <term>===><factor><N2>

    FIRST: {TK_ID, TK_NUM, TK_RNUM, TK_OP}

    FOLLOW: {TK_CL, TK_SEM, TK_PLUS, TK_MINUS}

    As there is just one production this rule is already in LL(1).

37. <factor>===><var> | TK_OP <arithmeticExpression> TK_CL

    FIRST: {TK_ID, TK_NUM, TK_RNUM, TK_OP}

    FOLLOW: {TK_MUL, TK_DIV, TK_CL, TK_PLUS, TK_MINUS, TK_SEM}

    FIRST(<var>) = {TK_ID, TK_NUM, TK_RNUM}, FIRST(TK_OP) = {TK_OP}.

    This shows that FIRST(<var>) and FIRST(TK_OP) are disjoint

    This implies that this rule is in LL(1).

38. <N2>===><operator2><factor><N2> | ∈

    FIRST: {TK_MUL, TK_DIV, ∈}

    FOLLOW: {TK_CL, TK_SEM, TK_PLUS, TK_MINUS}

FIRST(<operator2>) = {TK_MUL, TK_DIV}, FIRST(∈) = {∈}, FOLLOW(<N2>) = {TK_CL, TK_SEM, TK_PLUS, TK_MINUS}.

This shows that FIRST(<operator2>) and FIRST(∈) are disjoint, and FOLLOW(<N2>) ∩ FIRST(<operator2>) = ϕ.

This implies that this rule is in LL(1).

39. <operator1> ===> TK_PLUS | TK_MINUS

FIRST: {TK_PLUS, TK_MINUS}

FOLLOW: {TK_OP, TK_ID, TK_NUM, TK_RNUM}

FIRST(TK_PLUS) = {TK_PLUS} and FIRST(TK_MINUS) = {TK_MINUS}.

This shows that FIRST(TK_PLUS) and FIRST(TK_MINUS) are disjoint.

This implies that this rule is in LL(1).

40. <operator2> ===> TK_MUL | TK_DIV

FIRST: {TK_MUL, TK_DIV}

FOLLOW: {TK_OP, TK_ID, TK_NUM, TK_RNUM}

FIRST(TK_MUL) = {TK_MUL} and FIRST(TK_DIV) = {TK_DIV}.

This shows that FIRST(TK_MUL) and FIRST(TK_DIV) are disjoint.

This implies that this rule is in LL(1).

41. <booleanExpression>===>TK_OP <booleanExpression> TK_CL <logicalOp> TK_OP <booleanExpression> TK_CL

FIRST: {TK_OP}

FOLLOW: {TK_CL}

42. <booleanExpression>===> <var> <relationalOp> <var>

    FIRST: {TK_ID, TK_NUM, TK_RNUM}

    FOLLOW: {TK_CL}

43. <booleanExpression>===> TK_NOT TK_OP <booleanExpression> TK_CL

    FIRST: {TK_NOT}

    FOLLOW: {TK_CL}

    FIRST(TK_NOT) = {TK_NOT}, FIRST(<var>) = {TK_ID, TK_NUM, TK_RNUM} and FIRST(TK_OP) = {TK_OP}.

    This shows that FIRST(TK_NOT), FIRST(<var>) and FIRST(TK_OP) are all disjoint.

    This implies that this rule is in LL(1).

44. <var>===> <singleOrRecId> | TK_NUM | TK_RNUM

    FIRST: {TK_ID, TK_NUM, TK_RNUM}

    FOLLOW: {TK_MUL, TK_DIV, TK_CL, TK_PLUS, TK_MINUS, TK_SEM, TK_LT, TK_LE, TK_EQ, TK_GT, TK_GE, TK_NE }

    FIRST(<singleOrRecId>) = {TK_ID}, FIRST(TK_NUM) = {TK_NUM}, FIRST(TK_RNUM) = {TK_RNUM}, FOLLOW: {TK_MUL, TK_DIV, TK_CL, TK_PLUS, TK_MINUS, TK_SEM, TK_LT, TK_LE, TK_EQ, TK_GT, TK_GE, TK_NE }.

    This shows that FIRST(<singleOrRecId>), FIRST(TK_NUM), FIRST(TK_RNUM) are all disjoint, FOLLOW(<var>) ∩ FIRST(TK_NUM) = ϕ and FOLLOW(<var>) ∩ FIRST(TK_RNUM) = ϕ.

    This implies that this rule is in LL(1).

45. <logicalOp>===>TK_AND | TK_OR

FIRST: {TK_AND, TK_OR}

FOLLOW: {TK_OP}

46. <relationalOp>===> TK_LT | TK_LE | TK_EQ |TK_GT | TK_GE | TK_NE

FIRST: {TK_LT, TK_LE, TK_EQ, TK_GT, TK_GE, TK_NE}

FOLLOW: {TK_ID, TK_NUM, TK_RNUM}

47. <returnStmt>===>TK_RETURN <optionalReturn> TK_SEM

FIRST: {TK_RETURN}

FOLLOW: {TK_END}

48. <optionalReturn>===>TK_SQL <idList> TK_SQR | ∈

FIRST: {TK_SQL, ∈}

FOLLOW: {TK_SEM}

49. <idList>===> TK_ID <more_ids>

FIRST: {TK_ID}

FOLLOW: {TK_SQR}

50. <more_ids>===> TK_COMMA <idList> | ∈

FIRST: {TK_COMMA, ∈}

FOLLOW: {TK_SQR}

51. <definetypestmt>===>TK_DEFINETYPE <A> TK_RUID TK_AS TK_RUID

    FIRST: {TK_DEFINETYPE}

    FOLLOW: {TK_RECORD, TK_UNION}

52. <A>==>TK_RECORD | TK_UNION

    FIRST: {TK_RECORD, TK_UNION}

    FOLLOW: {TK_RUID}