1. The grammar which describes the mini program language that we've got has some left recursion and common left prefixes which we need to eliminate before coding the parser in C. The corrections are:

* VAR\_DEFINITIONS has a left recursion.
  + - Correction:  
      VAR\_DEFINITIONS -> VAR\_DEFINITION VAR\_DEFINITIONS'

VAR\_DEFINITIONS' -> ; VAR\_DEFINITIONS |

* VARIABLES\_LIST has a left recursion.
  + Correction:   
    VARIABLES\_LIST -> VARIABLE VARIABLE\_LIST'  
    VARIABLES\_LIST' -> ,VARIABLE VARIABLE\_LIST |
* VARIABLE has a common left prefixes.
  + Correction:   
    VARIABLE -> id VARIABLE'.

VARIABLE' -> [int\_number] |

* FUNC\_DEFINITIONS has a left recursion.
  + Correction:   
    FUNC\_DEFINITIONS -> FUNC\_DEFINITION FUNC\_DEFINITIONS'.

FUNC\_DEFINITIONS' -> FUNC\_DEFINITION FUNC\_DEFINITION' |

* STATEMENTS has a common left prefixes.
  + - Correction:   
      STATEMENTS -> STATEMENT; STATEMENTS'

STATAMENTS' -> STATEMENTS |

* STATEMENT has a common left prefixes.
  + - Correction:  
      STATEMENT -> BLOCK | return RETURN\_STATEMENT' | id ID\_STATAMENT

RETURN\_STATEMENT' -> EXPRESSION |

ID\_STATAMENT -> VARAIBLE' = EXPRESSION | (PARAMETERS\_LIST)

* EXPRESSION has a common left prefixes.
  + Correction:   
    EXPRESSION -> int\_number | real\_number | id EXPRESSION'

EXPRESSION' -> VARIABLE' | ar\_op EXPRESSION

Therefore, the fixed grammar which suitable for LL(1) parser is:

1. PROGRAM -> program VAR\_DEFINITIONS; STATEMENTS end FUNC\_DEFINITIONS
2. VAR\_DEFINITIONS -> VAR\_DEFINITION VAR\_DEFINITIONS'
3. VAR\_DEFINITIONS' -> ; VAR\_DEFINITIONS |
4. VAR\_DEFINITION -> TYPE VARIABLES\_LIST
5. TYPE -> real | integer
6. VARIABLES\_LIST -> VARIABLE VARIABLE\_LIST'
7. VARIABLES\_LIST' -> ,VARIABLE VARIABLE\_LIST' |
8. VARIABLE -> id VARIABLE'
9. VARIABLE' -> [int\_number] *|*
10. FUNC\_DEFINITIONS -> FUNC\_DEFINITION FUNC\_DEFINITIONS'
11. FUNC\_DEFINITIONS' -> FUNC\_DEFINITION FUNC\_DEFINITION' |
12. FUNC\_DEFINITION -> RETURNED\_TYPE id (PARAM\_DEFINITIONS)   
     BLOCK
13. RETURNED\_TYPE -> void | TYPE
14. PARAM\_DEFINITIONS -> VAR\_DEFINITIONS *|*
15. STATEMENTS -> STATEMENT; STATEMENTS'
16. STATAMENTS' -> STATEMENTS |
17. STATEMENT -> BLOCK | return RETURN\_STATEMENT' | id ID\_STATAMENT'
18. RETURN\_STATEMENT' -> EXPRESSION |
19. ID\_STATAMENT' -> VARAIBLE' = EXPRESSION | (PARAMETERS\_LIST)
20. BLOCK -> { VAR\_DEFINITIONS; STATEMENTS }
21. PARAMETERS\_LIST -> VARIABLES\_LIST *|*
22. EXPRESSION -> int\_number | real\_number | id EXPRESSION'
23. EXPRESSION' -> VARIABLE' | ar\_op EXPRESSION

**Note1:** the variable FUNCTION\_CALL is unnecessary after the elimination.

1. Calculation of attributes Nullable, First and Follow.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **The variable / attribute** | **Nullable** | **First** | **Follow** |
| 1 | **PROGRAM** | - | {program} | {EOF} |
| 2 | **VAR\_DEFINITIONS** | - | {real integer} | {; )} |
| 3 | **VAR\_DEFINITIONS'** | + | {;} | {; )} |
| 4 | **VAR\_DEFINITION** | - | {real integer} | {; )} |
| 5 | **TYPE** | - | {real integer} | {id} |
| 6 | **VARIABLES\_LIST** | - | {id} | {; )} |
| 7 | **VARIABLES\_LIST'** | + | {,} | {; )} |
| 8 | **VARIABLE** | - | {id} | {, ; )} |
| 9 | **VARIABLE'** | + | {[} | {= , ; )} |
| 10 | **FUNC\_DEFINITIONS** | - | {void real integer} | {EOF} |
| 11 | **FUNC\_DEFINITIONS'** | + | {void real integer} | {EOF} |
| 12 | **FUNC\_DEFINITION** | - | {void real integer} | {void real integer EOF} |
| 13 | **RETURNED\_TYPE** | - | {void real integer} | {id} |
| 14 | **PARAM\_DEFINITIONS** | + | {real integer} | {)} |
| 15 | **STATEMENTS** | - | {return id { } | {end } } |
| 16 | **STATEMENTS'** | + | {return id { } | {end } } |
| 17 | **STATEMENT** | - | {return id { } | {;} |
| 18 | **RETURN\_STATEMENT'** | + | {int\_number real\_number id} | {;} |
| 19 | **ID\_STATEMENT'** | - | {( [ = } | {;} |
| 20 | **BLOCK** | - | { { } | {void real integer EOF ;} |
| 21 | **PARAMETERS\_LIST** | + | {id} | {)} |
| 22 | **EXPRESSION** | - | {int\_number real\_number id} | {;} |
| 23 | **EXPRESSION'** | + | {ar\_op [ } | {;} |

**Note1:** we have made the predictive table to easy the coding part. It attached in Excel file in the ZIP.

**Note2:** in the table, the variable VAR\_DEFINITIONS' with token ';' includes two rules, so we have to look ahead 2 tokens.