**left recursion**

1. Program →Start-Symbols ClassDeclaration End-Symbols.

2. Start-Symbols →@| ^

3. End-Symbols→$ |#

4. ClassDeclaration→ Type ID{ Class\_Implementation} | Type ID Infer { Class\_Implementation}

5. Class\_Implementation→ Variable\_Decl Class\_Implementation| Method\_Decl Class\_Implementation | Comment Class\_Implementation | require\_command Class\_Implementation| Func \_Call

6. Method\_Decl→ Func Decl ;| Func Decl { Variable\_Decl Statements }

7. Func Decl →Type ID (ParameterList)

8. Type → Ipok |Sipok |Craf |Sequence |Ipokf |Sipokf |Valueless |Rational , {ID}

9. ParameterList →em| None | Non-Empty List {NA}

10. Non-Empty List→ Type ID Non-Empty List’ | Non-Empty List , Type ID

Type ID Non-Empty List’ = , Type ID Type ID Non-Empty List’ |em

11. Variable\_Decl→ em | Type ID\_List ; Variable\_Decl | Type ID\_List [ID] ; Variable\_Decl

12. ID\_List →ID ID\_List’| ID\_List , ID

ID\_List’= , ID ID\_List’ |em

13. Statements→em Statements’

Statements’ = Statement Statements’|em

14. Statement→Assignment | If \_Statement | However \_Statement | when\_Statement | Respondwith \_ Statement | Endthis \_Statement|Scanvalur (ID ); | Print (Expression); |

15. Assignment→ Variable\_Decl = Expression;

16. Func \_Call → ID (Argument\_List) ;

17. Argument\_List →em | NonEmpty\_Argument\_List

18. NonEmpty\_Argument\_List →Expression NonEmpty\_Argument\_List’

Expression NonEmpty\_Argument\_List’ = , Expression Expression NonEmpty\_Argument\_List’|em

19. Block Statements→{ statements }

20. If \_Statement→ if (Condition \_Expression) Block Statements | if (Condition \_Expression) Block Statements else Block Statements

21. Condition \_Expression→ Condition |Condition Condition \_Op Condition

22. Condition \_Op → && | ||

23. Condition→ Expression Comparison \_Op Expression

24. Comparison \_Op → == | != | > | >= | < | <=

25. However \_Statement → However (Condition \_Expression) Block Statements

26. when \_Statement → when ( expression ; expression ; expression ) Block Statements

27. Respondwith \_Statement→ Respondwith Expression ; | return ID ;

28. Endthis \_Statement→ Endthis;

29. Expression → Term Expression’

Expression’= Add\_Op Term Expression’ |em

30. Add\_Op → + | -

31. Term→Factor| Term Mul\_Op Factor

32. Mul\_Op→\* | /

33. Factor→ ID| Number

34. Comment → | \*\*\*STR

35. Require\_command →Require(F\_name.txt);

36. F\_name →STR