|  |  |
| --- | --- |
| Original Grammar:  program ::= program expr '\n'  |  expr ::= NUMBER  | '(' expr ')'  | expr '+' expr  | expr '-' expr  | expr '\*' expr  | expr '/' expr  | '-' expr  Grammar without left recursion and after left-factoring and removing program recursion:  program ::= expr '\n'  expr ::= term expr\_tail  expr\_tail::= '+' term expr\_tail  | '-' term expr\_tail  | ε  term ::= factor\_p term\_tail  term\_tail::= '\*' factor term\_tail  | '/' factor term\_tail  | ε  factor ::= NUMBER  | '(' expr ')'  factor\_p ::= '-' factor  | factor | FIRST(program) = {NUMBER - (}  FIRST(expr) = {NUMBER - (}  FIRST(expr\_tail) = {+ -}  FIRST(term) = {NUMBER – (}  FIRST(term\_tail) = {\* /}  FIRST(factor) = {NUMBER (}  FIRST(factor\_p) = {NUMBER - (}  FOLLOW(program) = { }  FOLLOW(expr) = {\n )}  FOLLOW(expr\_tail) = {\n )}  FOLLOW(term) = {+ - ) \n}  FOLLOW(term\_tail) = {+ - ) \n}  FOLLOW(factor) = {\* / + - ) \n}  FOLLOW(factor\_p) = {\* / + - ) \n} |