**COMPILTER CONSTRUCTIONS**

**Grammer**

After Eliminating Left Recurssion.

<program> 🡪 <data decls> <func list>   
<func list> 🡪 **empty**| <func> <func list>   
<func> 🡪 <func decl> **semicolon** | <func decl> **left\_brace**<data decls> <statements>**right\_brace**   
<func decl> 🡪 <type name> **ID** **left\_parenthesis**<parameter list> **right\_parenthesis**   
<type name> 🡪 **int** | **void | binary | decimal**   
<parameter list> 🡪 **empty** | **void** | <non-empty list>

<non-empty list> 🡪 **int** **ID** <non-empty-list>` | **void** **ID** <non-empty-list>` | **binary** **ID** <non-empty-list>` | **decimal** **ID** <non-empty-list>`

<non-empty-list>`🡪 **empty** | **comma** <type name> **ID** <non-empty-list>`

<data decls> 🡪 **empty** | **int** <id list> **semicolon** <data decls> | **void** <id list> **semicolon** <data decls> | **binary** <id list> **semicolon** <data decls> | **decimal** <id list> **semicolon** <data decls>

<id list> 🡪 <id> <id list>`

<id list>` 🡪 **empty** | **comma** <id> <id list>`

<id> 🡪 **ID** | **ID** **left\_bracket** <expression>**right\_bracket**

<block statements> 🡪 **left\_brace**<statements> **right\_brace**

<statements> 🡪 **empty** | <statement> <statements>   
<statement> 🡪 <assignment>| <func call>| <if statement> | <while statement> | <return statement> | <break statement> | <continue statement> | **read left\_parenthesis  ID right\_parenthesis semicolon**| **write left\_parenthesis**<expression> **right\_parenthesis semicolon** | **print left\_parenthesis  STRING right\_parenthesis semicolon**

<assignment> 🡪 **ID** **equal\_sign** <expression> **semicolon** | **ID** **left\_bracket** <expression>**right\_bracket** **equal\_sign** <expression> **semicolon**

<func call> 🡪 **ID left\_parenthesis**<expr list> **right\_parenthesis semicolon**

<expr list> --> **empty** | <non-empty expr list>

<non-empty expr list> 🡪 <expression> <non-empty expr list>`

<non-empty expr list>` 🡪 **comma** <expression> <non-empty expr list>` | **empty**

<if statement> 🡪 **if** **left\_parenthesis**<condition expression>**right\_parenthesis**<block statements>   
<condition expression> 🡪<condition> | <condition> <condition op> <condition>   
<condition op> 🡪 **double\_and\_sign** | **double\_or\_sign**   
<condition> 🡪 <expression> <comparison op> <expression>   
<comparison op> 🡪 **==** | **!=** | **>** | **>= | < | <=**

<while statement> 🡪 **while left\_parenthesis**<condition expression> **right\_parenthesis** <block statements>   
<return statement> 🡪 **return** <expression> **semicolon | return semicolon**   
<break statement> 🡪 **break semicolon**   
**<**continue statement> 🡪 **continue semicolon**

<expression> 🡪 <term><expression>`

<expression>` 🡪 <addop> <term> <expression>` | **empty**

<addop> 🡪 **plus\_sign** | **minus\_sign**

<term> 🡪 <factor><term>`

<term>` 🡪 <mulop> <factor> <term>` | **empty**

<mulop> 🡪 **star\_sign** | **forward\_slash**

<factor> 🡪 **ID** | **ID left\_bracket**<expression> **right\_bracket** | **ID** **left\_parenthesis** <expr list> **right\_parenthesis** | **NUMBER | minus\_sign NUMBER** | **left\_parenthesis** <expression>**right\_parenthesis**

After Left Factoring

<program> 🡪 <data decls> <func list>   
<func list> 🡪 **empty**

| <func> <func list>   
<func> 🡪 <func decl> <func>`

<func>` 🡪 **semicolon**

| **left\_brace**<data decls> <statements> **right\_brace** 

<func decl> 🡪 <type name> **ID** **left\_parenthesis**<parameter list> **right\_parenthesis**   
<type name> 🡪 **int**

| **void**

**| binary**

**| decimal**

<parameter list> 🡪 **empty**

| **void**

| <non-empty list>

<non-empty list> 🡪 **int** **ID** <non-empty-list>`

| **void** **ID** <non-empty-list>`

| **binary** **ID** <non-empty-list>`

| **decimal** **ID** <non-empty-list>`

<non-empty-list>` 🡪 **empty**

| **comma** <type name> **ID** <non-empty-list>`

<data decls> 🡪 **empty**

| **int** <id list> **semicolon** <data decls>

| **void** <id list> **semicolon** <data decls>

| **binary** <id list> **semicolon** <data decls>

| **decimal** <id list> **semicolon** <data decls>

<id list> 🡪 <id> <id list>`

<id list>` 🡪 **empty**

| **comma** <id> <id list>`

<id> 🡪 **ID** <id>`

<id>` 🡪**empty**

**| left\_bracket** <expression>**right\_bracket**

<block statements> 🡪 **left\_brace**<statements> **right\_brace**

<statements> 🡪 **empty**

| <statement> <statements>

<statement> 🡪 <assignment>

| <func call>

| <if statement>

| <while statement>

| <return statement>

| <break statement>

| <continue statement>

| **read left\_parenthesis  ID right\_parenthesis semicolon**

| **write left\_parenthesis**<expression> **right\_parenthesis semicolon**

| **print left\_parenthesis  STRING right\_parenthesis semicolon**

<assignment> 🡪 **ID** <assignment>`

<assignment>` 🡪 **equal\_sign** <expression> **semicolon** | **left\_bracket** <expression>**right\_bracket** **equal\_sign** <expression> **semicolon**

<func call> 🡪 **ID left\_parenthesis**<expr list> **right\_parenthesis semicolon**

<expr list> --> **empty** | <non-empty expr list>

<non-empty expr list> 🡪 <expression> <non-empty expr list>`

<non-empty expr list>` 🡪 **comma** <expression> <non-empty expr list>` | **empty**

<if statement> 🡪 **if** **left\_parenthesis**<condition expression>**right\_parenthesis**<block statements>   
<condition expression> 🡪<condition><condition expression>`

<condition expression>`🡪 **empty** | <condition op> <condition>   
<condition op> 🡪 **double\_and\_sign** | **double\_or\_sign**   
<condition> 🡪 <expression> <comparison op> <expression>   
<comparison op> 🡪 **==** | **!=** | **>** | **>= | < | <=**

<while statement> 🡪 **while left\_parenthesis**<condition expression> **right\_parenthesis** <block statements>   
<return statement> 🡪 **return** <return statement>`

<return statement>` 🡪 <expression> **semicolon | semicolon** 

<break statement> 🡪 **break semicolon**   
**<**continue statement> 🡪 **continue semicolon**

<expression> 🡪 <term><expression>`

<expression>` 🡪 <addop> <term> <expression>` | **empty**

<addop> 🡪 **plus\_sign** | **minus\_sign**

<term> 🡪 <factor><term>`

<term>` 🡪 <mulop> <factor> <term>` | **empty**

<mulop> 🡪 **star\_sign** | **forward\_slash**

<factor> 🡪 **ID** <factor>` | **NUMBER | minus\_sign NUMBER** | **left\_parenthesis** <expression>**right\_parenthesis**

<factor>` 🡪 **empty | left\_bracket**<expression> **right\_bracket** | **left\_parenthesis** <expr list> **right\_parenthesis**

After Checking First+() conditions

<program> 🡪 <data decls> <func list>

<func list> 🡪 **empty**

| <func> <func list>

<func> 🡪 <func decl> <func>`

<func>` 🡪 **semicolon**

| **left\_brace**<data decls> <statements> **right\_brace** 

<func decl> 🡪 <type name> **ID** **left\_parenthesis**<parameter list> **right\_parenthesis**

<type name> 🡪 **int**

| **void**

**| binary**

**| decimal**

<parameter list> 🡪 **empty**

| **int** **ID** <non-empty-list>`

| **binary** **ID** <non-empty-list>`

| **decimal** **ID** <non-empty-list>`

| void <parameter list>`

<parameter list>` 🡪 **ID** <non-empty-list>`

| **empty**

<non-empty list> 🡪 **int** **ID** <non-empty-list>`

| **void** **ID** <non-empty-list>`

| **binary** **ID** <non-empty-list>`

| **decimal** **ID** <non-empty-list>`

<non-empty-list>` 🡪 **empty**

| **comma** <type name> **ID** <non-empty-list>`

<data decls> 🡪 **empty**

| **int** <id list> **semicolon** <data decls>

| **void** <id list> **semicolon** <data decls>

| **binary** <id list> **semicolon** <data decls>

| **decimal** <id list> **semicolon** <data decls>

<id list> 🡪 <id> <id list>`

<id list>` 🡪 **empty**

| **comma** <id> <id list>`

<id> 🡪 **ID** <id>`

<id>` 🡪**empty**

**| left\_bracket** <expression>**right\_bracket**

<block statements> 🡪 **left\_brace**<statements> **right\_brace**

<statements> 🡪 **empty**

| <statement> <statements>

<statement> 🡪 <assignment>

| <func call>

| <if statement>

| <while statement>

| <return statement>

| <break statement>

| <continue statement>

| **read left\_parenthesis  ID right\_parenthesis semicolon**

| **write left\_parenthesis**<expression> **right\_parenthesis semicolon**

| **print left\_parenthesis  STRING right\_parenthesis semicolon**

<assignment> 🡪 **ID** <assignment>`

<assignment>` 🡪 **equal\_sign** <expression> **semicolon** | **left\_bracket** <expression>**right\_bracket** **equal\_sign** <expression> **semicolon**

<func call> 🡪 **ID left\_parenthesis**<expr list> **right\_parenthesis semicolon**

<expr list> --> **empty** | <non-empty expr list>

<non-empty expr list> 🡪 <expression> <non-empty expr list>`

<non-empty expr list>` 🡪 **comma** <expression> <non-empty expr list>` | **empty**

<if statement> 🡪 **if** **left\_parenthesis**<condition expression>**right\_parenthesis**<block statements>   
<condition expression> 🡪<condition><condition expression>`

<condition expression>`🡪 **empty** | <condition op> <condition>   
<condition op> 🡪 **double\_and\_sign** | **double\_or\_sign**   
<condition> 🡪 <expression> <comparison op> <expression>   
<comparison op> 🡪 **==** | **!=** | **>** | **>= | < | <=**

<while statement> 🡪 **while left\_parenthesis**<condition expression> **right\_parenthesis** <block statements>   
<return statement> 🡪 **return** <return statement>`

<return statement>` 🡪 <expression> **semicolon | semicolon** 

<break statement> 🡪 **break semicolon**   
**<**continue statement> 🡪 **continue semicolon**

<expression> 🡪 <term><expression>`

<expression>` 🡪 <addop> <term> <expression>` | **empty**

<addop> 🡪 **plus\_sign** | **minus\_sign**

<term> 🡪 <factor><term>`

<term>` 🡪 <mulop> <factor> <term>` | **empty**

<mulop> 🡪 **star\_sign** | **forward\_slash**

<factor> 🡪 **ID** <factor>` | **NUMBER | minus\_sign NUMBER** | **left\_parenthesis** <expression>**right\_parenthesis**

<factor>` 🡪 **empty | left\_bracket**<expression> **right\_bracket** | **left\_parenthesis** <expr list> **right\_parenthesis**

Changing <program>

<program> 🡪 <type name> <program>`

| **empty**

<program>` 🡪 **ID** <program>``

<program>`` 🡪 **left\_parenthesis**<parameter list> **right\_parenthesis** <program>```

| **semicolon** <program>

| **comma** <id> <id list>` **semicolon** <program>

| **left\_bracket** <expression>**right\_bracket** program````

<program>``` 🡪 **semicolon** <func list>

| **left\_brace** <data decls><statements> **right\_brace** <func list>

<program>```` 🡪 **semicolon** <program>

| **comma** <id> <id list>` **semicolon** <program>

<func list> 🡪 **empty**

| <func> <func list>

<func> 🡪 <func decl> <func>`

<func>` 🡪 **semicolon**

| **left\_brace**<data decls> <statements> **right\_brace** 

<func decl> 🡪 <type name> **ID** **left\_parenthesis**<parameter list> **right\_parenthesis**

<type name> 🡪 **int**

| **void**

**| binary**

**| decimal**

<parameter list> 🡪 **empty**

| **int** **ID** <non-empty-list>`

| **binary** **ID** <non-empty-list>`

| **decimal** **ID** <non-empty-list>`

| void <parameter list>`

<parameter list>` 🡪 **ID** <non-empty-list>`

| **empty**

<non-empty list> 🡪 **int** **ID** <non-empty-list>`

| **void** **ID** <non-empty-list>`

| **binary** **ID** <non-empty-list>`

| **decimal** **ID** <non-empty-list>`

<non-empty-list>` 🡪 **empty**

| **comma** <type name> **ID** <non-empty-list>`

<data decls> 🡪 **empty**

| **int** <id list> **semicolon** <data decls>

| **void** <id list> **semicolon** <data decls>

| **binary** <id list> **semicolon** <data decls>

| **decimal** <id list> **semicolon** <data decls>

<id list> 🡪 <id> <id list>`

<id list>` 🡪 **empty**

| **comma** <id> <id list>`

<id> 🡪 **ID** <id>`

<id>` 🡪**empty**

**| left\_bracket** <expression>**right\_bracket**

<block statements> 🡪 **left\_brace**<statements> **right\_brace**

<statements> 🡪 **empty**

| <statement> <statements>

<statement> 🡪 **ID** <statement>`

| <if statement>

| <while statement>

| <return statement>

| <break statement>

| <continue statement>

| **read left\_parenthesis ID right\_parenthesis semicolon**

| **write left\_parenthesis**<expression> **right\_parenthesis semicolon**

| **print left\_parenthesis STRING right\_parenthesis semicolon**

<statement>` 🡪 <assignment>`

| **left\_parenthesis**<expr list> **right\_parenthesis semicolon**

<assignment> 🡪 **ID** <assignment>`

<assignment>` 🡪 **equal\_sign** <expression> **semicolon**

| **left\_bracket** <expression>**right\_bracket** **equal\_sign** <expression> **semicolon**

<func call> 🡪 **ID left\_parenthesis**<expr list> **right\_parenthesis semicolon**

<expr list> 🡪 **empty**

| <non-empty expr list>

<non-empty expr list> 🡪 <expression> <non-empty expr list>`

<non-empty expr list>`🡪 **comma** <expression> <non-empty expr list>`

| **empty**

<if statement> 🡪 **if** **left\_parenthesis**<condition expression>**right\_parenthesis**<block statements> 

<condition expression> 🡪<condition><condition expression>`

<condition expression>`🡪 **empty**

| <condition op> <condition>

<condition op> 🡪 **double\_and\_sign**

| **double\_or\_sign**

<condition> 🡪 <expression> <comparison op> <expression> 

<comparison op> 🡪 **==**

| **!=**

| **>**

| **>=**

**| <**

**| <=**

<while statement> 🡪 **while left\_parenthesis**<condition expression> **right\_parenthesis** <block statements> 

<return statement> 🡪 **return** <return statement>`

<return statement>` 🡪 <expression> **semicolon**

**| semicolon** 

<break statement> 🡪 **break semicolon**   
**<**continue statement> 🡪 **continue semicolon**

<expression> 🡪 <term><expression>`

<expression>` 🡪 <addop> <term> <expression>`

| **empty**

<addop> 🡪 **plus\_sign**

| **minus\_sign**

<term> 🡪 <factor><term>`

<term>` 🡪 <mulop> <factor> <term>`

| **empty**

<mulop> 🡪 **star\_sign**

| **forward\_slash**

<factor> 🡪 **ID** <factor>`

| **NUMBER**

**| minus\_sign NUMBER**

| **left\_parenthesis** <expression>**right\_parenthesis**

<factor>` 🡪 **empty**

**| left\_bracket**<expression> **right\_bracket**

| **left\_parenthesis** <expr list> **right\_parenthesis**