Grammar:

```
// List of Colons

semi_colon_list
   :  ';'  semi_colon_list_tail

semi_colon_list_tail
   : semi_colon_list
   | ε

---

// Assignment Operators

assignment_operator //Code Gen = first. Add more later
   : '='
   | MUL_ASSIGN
   | DIV_ASSIGN
   | MOD_ASSIGN
   | ADD_ASSIGN
   | SUB_ASSIGN
   | LEFT_ASSIGN
   | RIGHT_ASSIGN
   | AND_ASSIGN
   | XOR_ASSIGN
   | OR_ASSIGN

---

//Data Types

typed_ID
   : type_specifier ID

type
   : CHAR
   | SHORT
   | INT
   | LONG
   | FLOAT
   | DOUBLE

type_specifier //TODO Const, Volatile later
   : type
   | SIGNED type
   | UNSIGNED type

```

```
47 ---
48
49 //Type List
50
51 type_specifier_list
52    :  type_specifier type_specifier_list_tail
53
54 type_specifier_list_tail
55    :  ε
56    | ',' type_specifier_list
57
58 ---
59
60 //Parameter List
61
62 parameter_specifier_list
63    :  type_specifier ID parameter_specifer_tail
64
65 parameter_specifer_list_tail
66    : ε
67    | ','  parameter_specifier_list
68
69 ---
70
71 //Program
72
73 program
74    :  body EOF
75
76 ---
77
78 //Body
79
80 body_statement
81    : statement body_statement_tail
82
83 body_statement_tail
84    : ε
85    : body
86
87 body_direct_declaration
88    : direct_declaration body_direct_declaration_tail
89
90 body_direct_declaration_tail
91    : ε
92    : body
93
```

```
94  body_function_declaration
95     : function_declaration body_function_declaration_tail
96
97  body_function_declaration_tail
98     : ε
99     : body
100
101 body_function_prototype
102    : function_declaration body_function_prototype_tail
103
104 body_function_prototype_tail
105    : ε
106    : body
107
108 body
109    | body_direct_declaration
110    | body_function_declaration
111    | body_function_prototype
112
113 ---
114
115 //Function Declaration & Direct Variable Declaration
116
117 typed_ID_common_prefix
118    : typed_ID typed_ID_tail
119    | VOID '(' function_prefix
120
121 typed_ID_tail:
122    : '(' function_prefix
123    | //TODO Variable stuff
124
125 function_prefix
126    :  typed_ID
127    |  VOID '('
128
129 function_prototype
130    :  function_prefix type_specifier_list ')' semi_colon_list
131    |  function_declaration_prefix semi_colon_list
132
133 function_declaration
134    :  function_prefix parameter_specifier_list ')' '{' statement_list '}'
135
136 function_declaration_tail
137    |  semi_colon_list
138    |  ε
139
140 ---
```

```
141
142 //Direct Declaration
143
144 direct_declaration
145    : type_ID semi_colon_list
146    | SIGNED type ID semi_colon_list
147    | UNSIGNED type ID semi_colon_list
148    | VOID ID semi_colon_list
149    | CHAR ID '=' STRING_LITERAL semi_colon_list
150
151 ---
152
153 //Statement and Statement List
154
155 statement
156    : compound_statement //Sub statements to loops, conditional, and code blocks
157    | expression_statement //Assignment, Boolean, Arithmetic Expressions
158    | selection_statement //IF Statements
159    | iteration_statement  //Loops
160    | semi_colon_list // End of statement one or more ;
161    | direct_declaration
162
163 statement_list
164    : statement statement_list_tail
165
166 statement_list_tail
167    : statment_list
168    | ε
169
170
171 compound_statement
172    : '{' compound_statement_tail
173
174
175 compound_statement_tail
176    : '}'
177    | statement_list '}'
178
179
180 expression_statement
181    : ';'
182    | expression ';'
183
184 //TODO Add Expression
185 expression
186
187 selection_statement
```

```
188     : IF '(' expression ')' statement
189     | IF '(' expression ')' statement ELSE statement
190
191
192
193 iteration_statement
194     : WHILE '(' expression ')' statement
195     | FOR '(' expression_statement expression_statement ')' statement
196     | FOR '(' expression_statement expression_statement expression ')' statement
```

$$program \Rightarrow body \;\; EOF \tag{1}$$

$$\begin{aligned} body \Rightarrow\; & stmt \\ & \mid stmt \;\; body \end{aligned} \tag{2}$$

$$\begin{aligned} stmt \Rightarrow\; & ID \;\; '=' \;\; expr \;\; ';' \\ & \mid IF \;\; '(' \;\; bexp \;\; ')' \;\; stmt \\ & \mid WHILE \;\; '(' \;\; bexp \;\; ')' \;\; stmt \\ & \mid '\{' \;\; substmts \\ & \mid ';' \end{aligned} \tag{3}$$

$$\begin{aligned} substmts \Rightarrow\; & stmt \;\; '\}' \\ & \mid stmt \;\; substmts \\ & \mid '\}' \end{aligned} \tag{4}$$

$$assignment \Rightarrow\; ID \;\; '=' \;\; expression \;\; ';' \tag{5}$$

$$\begin{aligned} expression \Rightarrow\; & expr \\ & \mid bexpr \end{aligned} \tag{6}$$

Notes - Continue down this route
TODO

- expr - arithmetic expressions. Make sure to get precedence (greater precedence last) and associativity correct. From `http://pages.cs.wisc.edu/~fischer/cs536.s08/course.hold/html/NOTES/3.CFG.html#assoc` Remove POW

```
exp      --> exp PLUS term        |  exp MINUS term      |  term
term     --> term TIMES factor    |  term DIVIDE factor  |  factor
```

```
factor   --> exponent POW factor  |  exponent
exponent --> MINUS exponent       |  final
final    --> INTLITERAL           |  LPAREN exp RPAREN
```

- bexpr - boolean expressions

```
bexp --> TRUE
bexp --> FALSE
bexp --> bexp OR bexp
bexp --> bexp AND bexp
bexp --> NOT bexp
bexp --> LPAREN bexp RPAREN
```

- stmt - add if and while loops to it. Need to make sure no ambiguity in control statements.