CONTEXT FREE GRAMMAR

1.	<program>	→	<global_declaration> spawn void base() { <base_prod> } <user_function>
2.	<global_declaration>	→	<global_var> <global_declaration>
3.	<global_declaration>	→	<global_comp> <global_declaration>
4.	<global_declaration>	→	<global_tower> <global_declaration>
5.	<global_declaration>	→	λ
6.	<global_var>	→	inter <gv_inter>;
7.	<global_var>	→	bloat <gv_bloat>;
8.	<global_var>	→	ping <gv_ping>;
9.	<global_var>	→	pool <gv_pool>;
10.	<gv_inter>	→	Identifier <gv_inter_tail>
11.	<gv_inter_tail>	→	= InterLiteral <add_gv_inter_tail>
12.	<gv_inter_tail>	→	<G_inter_array_dec>
13.	<gv_inter_tail>	→	λ
14.	<G_inter_array_dec>	→	[InterLiteral] <G_inter_1D_tail>
15.	<G_inter_1D_tail>	→	= { <G_inter_element> <G_add_inter_1D> }
16.	<G_inter_1D_tail>	→	[InterLiteral] <G_inter_2D_tail>
17.	<G_inter_1D_tail>	→	λ
18.	<G_inter_element>	→	InterLiteral
19.	<G_add_inter_1D>	→	, <G_inter_element> <G_add_inter_1D>
20.	<G_add_inter_1D>	→	λ
21.	<G_inter_2D_tail>	→	= { { <G_inter_element> <G_add_inter_1D> } <G_add_inter_2D> }
22.	<G_inter_2D_tail>	→	λ
23.	<G_add_inter_2D>	→	, {<G_inter_element> <G_add_inter_1D>} <G_add_inter_2D>
24.	<G_add_inter_2D>	→	λ

25.	<add_gv_inter_tail>	→	, Identifier <add_gv_inter_val_tail>
26.	<add_gv_inter_tail>	→	λ
27.	<add_gv_inter_val_tail>		= InterLiteral <add_gv_inter_tail>
28.	<add_gv_inter_val_tail>		λ
29.	<gv_bloat>	→	Identifier <gv_bloat_tail>
30.	<gv_bloat_tail>	→	= BloatLiteral <add_gv_bloat_tail>
31.	<gv_bloat_tail>	→	λ
32.	<gv_bloat_tail>	→	<G_bloat_array_dec>
33.	<G_bloat_array_dec>	→	[InterLiteral] <G_bloat_1D_tail>
34.	<G_bloat_1D_tail>	→	= { <G_bloat_element> <G_add_bloat_1D> }
35.	<G_bloat_1D_tail>	→	[InterLiteral] <G_bloat_2D_tail>
36.	<G_bloat_1D_tail>	→	λ
37.	<G_bloat_element>	→	BloatLiteral
38.	<G_add_bloat_1D>	→	, <G_bloat_element> <G_add_bloat_1D>
39.	<G_add_bloat_1D>	→	λ
40.	<G_bloat_2D_tail>	→	= { { <G_bloat_element> <G_add_bloat_1D> } <G_add_bloat_2D> }
41.	<G_bloat_2D_tail>	→	λ
42.	<G_add_bloat_2D>	→	, { <G_bloat_element> <G_add_bloat_1D> } <G_add_bloat_2D>
43.	<G_add_bloat_2D>	→	λ
44.	<add_gv_bloat_tail>	→	, Identifier <add_gv_bloat_val_tail>
45.	<add_gv_bloat_tail>	→	λ
46.	<add_gv_bloat_val_tail>		= BloatLiteral <add_gv_bloat_tail>
47.	<add_gv_bloat_val_tail>		λ
48.	<gv_ping>	→	Identifier <gv_ping_tail>
49.	<gv_ping_tail>	→	= PingLiteral <add_gv_ping_tail>
50.	<gv_ping_tail>	→	λ

51.	<gv_ping_tail>	→	<G_ping_array_dec>
52.	<G_ping_array_dec>	→	[InterLiteral] <G_ping_1D_tail>
53.	<G_ping_1D_tail>	→	= { <G_ping_element> <G_add_ping_1D> }
54.	<G_ping_1D_tail>	→	[InterLiteral] <G_ping_2D_tail>
55.	<G_ping_1D_tail>	→	λ
56.	<G_ping_element>	→	PingLiteral
57.	<G_add_ping_1D>	→	, <G_ping_element> <G_add_ping_1D>
58.	<G_add_ping_1D>	→	λ
59.	<G_ping_2D_tail>	→	= { { <G_ping_element> <G_add_ping_1D> } <G_add_ping_2D> }
60.	<G_ping_2D_tail>	→	λ
61.	<G_add_ping_2D>	→	, { <G_ping_element> <G_add_ping_1D> } <G_add_ping_2D>
62.	<G_add_ping_2D>	→	λ
63.	<add_gv_ping_tail>	→	, Identifier <add_gv_ping_val_tail>
64.	<add_gv_ping_tail>	→	λ
65.	<add_gv_ping_val_tail>		= PingLiteral <add_gv_ping_tail>
66.	<add_gv_ping_val_tail>		λ
67.	<gv_pool>	→	Identifier <gv_pool_tail>
68.	<gv_pool_tail>	→	= <pool_literal> <add_gv_pool_tail>
69.	<gv_pool_tail>	→	λ
70.	<gv_pool_tail>	→	<G_pool_array_dec>
71.	<G_pool_array_dec>	→	[InterLiteral] <G_pool_1D_tail>
72.	<G_pool_1D_tail>	→	= { <G_pool_element> <G_add_pool_1D> }
73.	<G_pool_1D_tail>	→	[InterLiteral] <G_pool_2D_tail>
74.	<G_pool_1D_tail>	→	λ
75.	<G_pool_element>	→	<pool_literal>
76.	<G_add_pool_1D>	→	, <G_pool_element> <G_add_pool_1D>

77.	<G_add_pool_1D>	→	λ
78.	<G_pool_2D_tail>	→	= { { <G_pool_element> <G_add_pool_1D> } <G_add_pool_2D> }
79.	<G_pool_2D_tail>	→	λ
80.	<G_add_pool_2D>	→	, { <G_pool_element> <G_add_pool_1D> } <G_add_pool_2D>
81.	<G_add_pool_2D>	→	λ
82.	<add_gv_pool_tail>	→	, Identifier <add_gv_pool_val_tail>
83.	<add_gv_pool_tail>	→	λ
84.	<add_gv_pool_val_tail>		= <pool_literal> <add_gv_pool_tail>
85.	<add_gv_pool_val_tail>		λ
86.	<pool_literal>	→	buff
87.	<pool_literal>	→	debuff
88.	<global_comp>	→	comp <gc_data_type>;
89.	<gc_data_type>	→	inter Identifier <gc_inter_tail>
90.	<gc_data_type>	→	bloat Identifier <gc_bloat_tail>
91.	<gc_data_type>	→	ping Identifier <gc_ping_tail>
92.	<gc_data_type>	→	pool Identifier <gc_pool_tail>
93.	<gc_inter_tail>	→	= InterLiteral <add_gc_inter_tail>
94.	<gc_inter_tail>	→	<gc_inter_array_dec>
95.	<add_gc_inter_tail>	→	, Identifier <add_gc_inter_val_tail>
96.	<add_gc_inter_tail>	→	λ
97.	<add_gc_inter_val_tail>		= InterLiteral <add_gc_inter_tail>
98.	<gc_inter_array_dec>	→	[InterLiteral] <gc_inter_1D_tail>
99.	<gc_inter_1D_tail>	→	= { <G_inter_element> <G_add_inter_1D> }
100.	<gc_inter_1D_tail>	→	[InterLiteral] <gc_inter_2D_tail>
101.	<gc_inter_2D_tail>	→	= { { <G_inter_element> <G_add_inter_1D>} <G_add_inter_2D> }
102.	<gc_bloat_tail>	→	= BloatLiteral <add_gc_bloat_tail>

103.	<gc_bloat_tail>	→	<gc_bloat_array_dec>
104.	<add_gc_bloat_tail>	→	, Identifier <add_gc_bloat_val_tail>
105.	<add_gc_bloat_tail>	→	λ
106.	<add_gc_bloat_val_tail>		= BloatLiteral <add_gc_bloat_tail>
107.	<gc_bloat_array_dec>	→	[InterLiteral] <gc_bloat_1D_tail>
108.	<gc_bloat_1D_tail>	→	= { <G_bloat_element> <G_add_bloat_1D> }
109.	<gc_bloat_1D_tail>	→	[InterLiteral] <gc_bloat_2D_tail>
110.	<gc_bloat_2D_tail>	→	= { { <G_bloat_element> <G_add_bloat_1D> } <G_add_bloat_2D> }
111.	<gc_ping_tail>	→	= PingLiteral <add_gc_ping_tail>
112.	<gc_ping_tail>	→	<gc_ping_array_dec>
113.	<add_gc_ping_tail>	→	, Identifier <add_gc_ping_val_tail>
114.	<add_gc_ping_tail>	→	λ
115.	<add_gc_ping_val_tail>		= PingLiteral <add_gc_ping_tail>
116.	<gc_ping_array_dec>	→	[InterLiteral] <gc_ping_1D_tail>
117.	<gc_ping_1D_tail>	→	= { <G_ping_element> <G_add_ping_1D> }
118.	<gc_ping_1D_tail>	→	[InterLiteral] <gc_ping_2D_tail>
119.	<gc_ping_2D_tail>	→	= { { <G_ping_element> <G_add_ping_1D>} <G_add_ping_2D> }
120.	<gc_pool_tail>	→	= <pool_literal> <add_gc_pool_tail>
121.	<gc_pool_tail>	→	<gc_pool_array_dec>
122.	<add_gc_pool_tail>	→	, Identifier <add_gc_pool_val_tail>
123.	<add_gc_pool_tail>	→	λ
124.	<add_gc_pool_val_tail>		= <pool_literal> <add_gc_pool_tail>
125.	<gc_pool_array_dec>	→	[InterLiteral] <gc_pool_1D_tail>
126.	<gc_pool_1D_tail>	→	= { <G_pool_element> <G_add_pool_1D> }
127.	<gc_pool_1D_tail>	→	[InterLiteral] <gc_pool_2D_tail>
128.	<gc_pool_2D_tail>	→	= { { <G_pool_element> <G_add_pool_1D> } <G_add_pool_2D> }

129.	<global_tower>	→	tower Identifier {{ <tower_var> }}
130.	<tower_var>	→	<gt_data_type><optional_array> Identifier; <add_tower_var>
131.	<add_tower_var>	→	<tower_var>
132.	<add_tower_var>	→	λ
133.	<gt_data_type>	→	inter
134.	<gt_data_type>	→	bloat
135.	<gt_data_type>	→	ping
136.	<gt_data_type>	→	pool
137.	<base_prod>	→	<local_declaration> <base_prod>
138.	<base_prod>	→	<statement> <base_prod>
139.	<base_prod>	→	λ
140.	<local_declaration>	→	<local_var>
141.	<local_declaration>	→	<local_comp>
142.	<local_declaration>	→	<local_tower>
143.	<local_var>	→	inter <lv_inter>;
144.	<local_var>	→	bloat <lv_bloat>;
145.	<local_var>	→	ping <lv_ping>;
146.	<local_var>	→	pool <lv_pool>;
147.	<lv_inter>	→	Identifier <lv_inter_tail>
148.	<lv_inter_tail>	→	= <lv_inter_value> <add_lv_inter_tail>
149.	<lv_inter_tail>	→	<L_inter_array_dec>
150.	<lv_inter_tail>	→	λ
151.	<lv_inter_value>	→	<math_expression>
152.	<math_expression>	→	<math_operand><math_tail_expression>
153.	<math_operand>	→	(<math_expression>)
154.	<math_operand>	→	inter(<inter_conversion_value>)
155.	<math_operand>	→	bloat(<bloat_conversion_value>)

156.	<math_operand>	→	InterLiteral
157.	<math_operand>	→	BloatLiteral
158.	<math_operand>	→	Identifier<value_type>
159.	<math_tail_expression>	→	<math_operator><math_operand><math_tail_expression>
160.	<math_tail_expression>	→	λ
161.	<math_operator>	→	+
162.	<math_operator>	→	-
163.	<math_operator>	→	*
164.	<math_operator>	→	/
165.	<math_operator>	→	%
166.	<inter_conversion_value>	→	PingLiteral
167.	<inter_conversion_value>	→	<math_expression>
168.	<inter_conversion_value>	→	hold()
169.	<value_type>	→	[<index_value>] <2D_index_value>
170.	<value_type>	→	.Identifier
171.	<value_type>	→	(<argument>)
172.	<value_type>	→	λ
173.	<index_value>	→	<math_expression>
174.	<2D_index_value>	→	[<index_value>]
175.	<2D_index_value>		λ
176.	<argument>	→	<literal_value> <additional_args>
177.	<argument>	→	Identifier<value_type> <additional_args>
178.	<argument>	→	<builtin_func_call> <additional_args>
179.	<argument>	→	λ
180.	<literal_value>	→	InterLiteral
181.	<literal_value>	→	BloatLiteral
182.	<literal_value>	→	PingLiteral

183.	<literal_value>	→	<pool_literal>
184.	<additional_args>	→	, <argument>
185.	<additional_args>	→	λ
186.	<builtin_func_call>	→	inter(<inter_conversion_value>)
187.	<builtin_func_call>	→	bloat(<bloat_conversion_value>)
188.	<builtin_func_call>	→	pool(<pool_conversion_value>)
189.	<builtin_func_call>	→	ping(<ping_conversion_value>)
190.	<L_inter_array_dec>	→	[<index_value>] <L_inter_1D_tail>
191.	<L_inter_1D_tail>	→	= { <L_inter_element> <L_add_inter_1D> }
192.	<L_inter_1D_tail>	→	[<index_value>] <L_inter_2D_tail>
193.	<L_inter_1D_tail>	→	λ
194.	<L_inter_element>	→	<lv_inter_value>
195.	<L_add_inter_1D>	→	, <L_inter_element> <L_add_inter_1D>
196.	<L_add_inter_1D>	→	λ
197.	<L_inter_2D_tail>	→	= { { <L_inter_element> <L_add_inter_1D>} <L_add_inter_2D> }
198.	<L_inter_2D_tail>	→	λ
199.	<L_add_inter_2D>	→	, { <L_inter_element> <L_add_inter_1D>} <L_add_inter_2D>
200.	<L_add_inter_2D>	→	λ
201.	<add_lv_inter_tail>	→	, Identifier <add_lv_inter_val_tail>
202.	<add_lv_inter_tail>	→	λ
203.	<add_lv_inter_val_tail>		= <lv_inter_value> <add_lv_inter_tail>
204.	<add_lv_inter_val_tail>		λ
205.	<lv_bloat>	→	Identifier <lv_bloat_tail>
206.	<lv_bloat_tail>	→	= <lv_bloat_value> <add_lv_bloat_tail>
207.	<lv_bloat_tail>	→	<L_bloat_array_dec>
208.	<lv_bloat_tail>	→	λ

209.	<lv_bloat_value>	→	<math_expression>
210.	<bloat_conversion_value>	→	PingLiteral
211.	<bloat_conversion_value>	→	<math_expression>
212.	<bloat_conversion_value>	→	hold()
213.	<L_bloat_array_dec>	→	[<index_value>] <L_bloat_1D_tail>
214.	<L_bloat_1D_tail>	→	= { <L_bloat_element> <L_add_bloat_1D> }
215.	<L_bloat_1D_tail>	→	[<index_value>] <L_bloat_2D_tail>
216.	<L_bloat_1D_tail>	→	λ
217.	<L_bloat_element>	→	<lv_bloat_value>
218.	<L_add_bloat_1D>	→	, <L_bloat_element> <L_add_bloat_1D>
219.	<L_add_bloat_1D>	→	λ
220.	<L_bloat_2D_tail>	→	= { { <L_bloat_element> <L_add_bloat_1D> } <L_add_bloat_2D> }
221.	<L_bloat_2D_tail>	→	λ
222.	<L_add_bloat_2D>	→	, { <L_bloat_element> <L_add_bloat_1D>} <L_add_bloat_2D>
223.	<L_add_bloat_2D>	→	λ
224.	<add_lv_bloat_tail>	→	, Identifier <add_lv_bloat_val_tail>
225.	<add_lv_bloat_tail>	→	λ
226.	<add_lv_bloat_val_tail>		= <lv_bloat_value> <add_lv_bloat_tail>
227.	<add_lv_bloat_val_tail>		λ
228.	<lv_ping>	→	Identifier <lv_ping_tail>
229.	<lv_ping_tail>	→	= <lv_ping_value> <add_lv_ping_tail>
230.	<lv_ping_tail>	→	<L_ping_array_dec>
231.	<lv_ping_tail>	→	λ
232.	<lv_ping_value>	→	ping(<ping_conversion_value>)
233.	<lv_ping_value>	→	hold()
234.	<lv_ping_value>	→	<string_concat>

235.	<ping_conversion_value>	→	InterLiteral
236.	<ping_conversion_value>	→	BloatLiteral
237.	<ping_conversion_value>	→	<pool_literal>
238.	<ping_conversion_value>	→	<string_concat>
239.	<string_concat>	→	<string_value> <string_tail_concat>
240.	<string_value>	→	Identifier<value_type>
241.	<string_value>	→	PingLiteral
242.	<string_tail_concat>	→	+ <string_concat>
243.	<string_tail_concat>	→	λ
244.	<L_ping_array_dec>	→	[<index_value>] <L_ping_1D_tail>
245.	<L_ping_1D_tail>	→	= { <L_ping_element> <L_add_ping_1D> }
246.	<L_ping_1D_tail>	→	[<index_value>] <L_ping_2D_tail>
247.	<L_ping_1D_tail>	→	λ
248.	<L_ping_element>	→	<lv_ping_value>
249.	<L_add_ping_1D>	→	, <L_ping_element> <L_add_ping_1D>
250.	<L_add_ping_1D>	→	λ
251.	<L_ping_2D_tail>	→	= { { <L_ping_element> <L_add_ping_1D>} <L_add_ping_2D> }
252.	<L_ping_2D_tail>	→	λ
253.	<L_add_ping_2D>	→	, { <L_ping_element> <L_add_ping_1D>} <L_add_ping_2D>
254.	<L_add_ping_2D>	→	λ
255.	<add_lv_ping_tail>	→	, Identifier <add_lv_ping_val_tail>
256.	<add_lv_ping_tail>	→	λ
257.	<add_lv_ping_val_tail>		= <lv_ping_value> <add_lv_ping_tail>
258.	<add_lv_ping_val_tail>		λ
259.	<lv_pool>	→	Identifier <lv_pool_tail>
260.	<lv_pool_tail>	→	= <lv_pool_value> <add_lv_pool_tail>

261.	<lv_pool_tail>	→	<L_pool_array_dec>
262.	<lv_pool_tail>	→	λ
263.	<lv_pool_value>	→	<general_expression>
264.	<pool_conversion_value>	→	<pool_convert>
265.	<pool_conversion_value>	→	Identifier<value_type>
266.	<pool_conversion_value>	→	hold()
267.	<pool_convert>	→	PingLiteral
268.	<pool_convert>	→	<pool_literal>
269.	<general_expression>	→	<general_operand> <general_tail_expression>
270.	<general_operand>	→	(<general_expression>)
271.	<general_operand>	→	! <general_operand>
272.	<general_operand>	→	inter(<inter_conversion_value>)
273.	<general_operand>	→	bloat(<bloat_conversion_value>)
274.	<general_operand>	→	InterLiteral
275.	<general_operand>	→	BloatLiteral
276.	<general_operand>	→	PingLiteral
277.	<general_operand>	→	<pool_literal>
278.	<general_operand>	→	Identifier<value_type>
279.	<general_operand>	→	pool(<pool_conversion_value>)
280.	<general_operand>	→	ping(<ping_conversion_value>)
281.	<general_tail_expression>	→	<general_operator><general_operand> <general_tail_expression>
282.	<general_tail_expression>	→	λ
283.	<general_operator>	→	<math_operator>
284.	<general_operator>	→	&&
285.	<general_operator>	→	||
286.	<general_operator>	→	==
287.	<general_operator>	→	!=

288.	<general_operator>	→	>
289.	<general_operator>	→	<
290.	<general_operator>	→	>=
291.	<general_operator>	→	<=
292.	<L_pool_array_dec>	→	[<index_value>] <L_pool_1D_tail>
293.	<L_pool_1D_tail>	→	= { <L_pool_element> <L_add_pool_1D> }
294.	<L_pool_1D_tail>	→	[<index_value>] <L_pool_2D_tail>
295.	<L_pool_1D_tail>	→	λ
296.	<L_pool_element>	→	<lv_pool_value>
297.	<L_add_pool_1D>	→	, <L_pool_element> <L_add_pool_1D>
298.	<L_add_pool_1D>	→	λ
299.	<L_pool_2D_tail>	→	= { { <L_pool_element> <L_add_pool_1D>} <L_add_pool_2D> }
300.	<L_pool_2D_tail>	→	λ
301.	<L_add_pool_2D>	→	, { <L_pool_element> <L_add_pool_1D>} <L_add_pool_2D>
302.	<L_add_pool_2D>	→	λ
303.	<add_lv_pool_tail>	→	, Identifier <add_lv_pool_val_tail>
304.	<add_lv_pool_tail>	→	λ
305.	<add_lv_pool_val_tail>		= <lv_pool_value> <add_lv_pool_tail>
306.	<add_lv_pool_val_tail>		λ
307.	<local_comp>	→	comp <lc_data_type>;
308.	<lc_data_type>	→	inter Identifier <lc_inter_tail>
309.	<lc_data_type>	→	bloat Identifier <lc_bloat_tail>
310.	<lc_data_type>	→	ping Identifier <lc_ping_tail>
311.	<lc_data_type>	→	pool Identifier <lc_pool_tail>
312.	<lc_inter_tail>	→	= <lv_inter_value> <add_lc_inter_tail>
313.	<lc_inter_tail>	→	<lc_inter_array_dec>

314.	<add_lc_inter_tail>	→	, Identifier <add_lc_inter_val_tail>
315.	<add_lc_inter_tail>	→	λ
316.	<add_lc_inter_val_tail>		= <lv_inter_value> <add_lc_inter_tail>
317.	<lc_inter_array_dec>	→	[<index_value>] <lc_inter_1D_tail>
318.	<lc_inter_1D_tail>	→	= { <L_inter_element> <L_add_inter_1D> }
319.	<lc_inter_1D_tail>	→	[<index_value>] <lc_inter_2D_tail>
320.	<lc_inter_2D_tail>	→	= { { <L_inter_element> <L_add_inter_1D> } <L_add_inter_2D> }
321.	<lc_bloat_tail>	→	= <lv_bloat_value> <add_lc_bloat_tail>
322.	<lc_bloat_tail>	→	<lc_bloat_array_dec>
323.	<add_lc_bloat_tail>	→	, Identifier <add_lc_bloat_val_tail>
324.	<add_lc_bloat_tail>	→	λ
325.	<add_lc_bloat_val_tail>		= <lv_bloat_value> <add_lc_bloat_tail>
326.	<lc_bloat_array_dec>	→	[<index_value>] <lc_bloat_1D_tail>
327.	<lc_bloat_1D_tail>	→	= { <L_bloat_element> <L_add_bloat_1D> }
328.	<lc_bloat_1D_tail>	→	[<index_value>] <lc_bloat_2D_tail>
329.	<lc_bloat_2D_tail>	→	= { { <L_bloat_element> <L_add_bloat_1D> } <L_add_bloat_2D> }
330.	<lc_ping_tail>	→	= <lv_ping_value> <add_lc_ping_tail>
331.	<lc_ping_tail>	→	<lc_ping_array_dec>
332.	<add_lc_ping_tail>	→	, Identifier <add_lc_ping_val_tail>
333.	<add_lc_ping_tail>	→	λ
334.	<add_lc_ping_val_tail>		= <lv_ping_value> <add_lc_ping_tail>
335.	<lc_ping_array_dec>	→	[<index_value>] <lc_ping_1D_tail>
336.	<lc_ping_1D_tail>	→	= { <L_ping_element> <L_add_ping_1D> }
337.	<lc_ping_1D_tail>	→	[<index_value>] <lc_ping_2D_tail>
338.	<lc_ping_2D_tail>	→	= { { <L_ping_element> <L_add_ping_1D>} <L_add_ping_2D> }
339.	<lc_pool_tail>	→	= <lv_pool_value> <add_lc_pool_tail>

340.	<lc_pool_tail>	→	<lc_pool_array_dec>
341.	<add_lc_pool_tail>	→	, Identifier <add_lc_pool_val_tail>
342.	<add_lc_pool_tail>	→	λ
343.	<add_lc_pool_val_tail>		= <lv_pool_value> <add_lc_pool_tail>
344.	<lc_pool_array_dec>	→	[<index_value>] <lc_pool_1D_tail>
345.	<lc_pool_1D_tail>	→	= { <L_pool_element> <L_add_pool_1D> }
346.	<lc_pool_1D_tail>	→	[<index_value>] <lc_pool_2D_tail>
347.	<lc_pool_2D_tail>	→	= { { <L_pool_element> <L_add_pool_1D> } <L_add_pool_2D> }
348.	<local_tower>	→	tower Identifier Identifier;
349.	<statement>	→	Identifier<stm_type>;
350.	<statement>	→	<loop_stm>
351.	<statement>	→	<cond_stm>
352.	<statement>	→	push(<push_value>);
353.	<statement>		recall <recall_value>;
354.	<stm_type>	→	<assign_value_type> <assignment>
355.	<stm_type>	→	(<argument>)
356.	<assign_value_type>	→	[<index_value>] <2D_index_value>
357.	<assign_value_type>	→	.Identifier
358.	<assign_value_type>	→	λ
359.	<assignment>	→	= <assign_value>
360.	<assign_value>	→	hold()
361.	<assign_value>	→	<general_expression>
362.	<assign_value>	→	{ <1D_2D_array> }
363.	<1D_2D_array>	→	<assign_array_element> <add_assign_1D>
364.	<1D_2D_array>	→	{ <assign_array_element> <add_assign_1D> } <add_assign_2D>
365.	<assign_array_element>	→	<general_expression>

366.	<add_assign_1D>	→	, <assign_array_element> <add_assign_1D>
367.	<add_assign_1D>	→	λ
368.	<add_assign_2D>	→	, { <assign_array_element> <add_assign_1D> } <add_assign_2D>
369.	<add_assign_2D>	→	λ
370.	<loop_stm>	→	for <init> <for_keyword> <end> { <content> }
371.	<loop_stm>	→	while (<condition>) { <content> }
372.	<loop_stm>	→	do { <content> } while (<condition>)
373.	<init>	→	Identifier = <init_value>
374.	<init_value>		InterLiteral
375.	<init_value>		Identifier <value_type>
376.	<for_keyword>		up
377.	<for_keyword>		down
378.	<end>	→	InterLiteral
379.	<end>	→	Identifier <value_type>
380.	<content>	→	<local_declaration><content>
381.	<content>	→	<statement><content>
382.	<content>	→	<loop_terminator><content>
383.	<content>	→	λ
384.	<condition>	→	<general_expression>
385.	<loop_terminator>	→	destroy;
386.	<loop_terminator>	→	commit;
387.	<cond_stm>	→	if (<condition>) <body> <else_clause>
388.	<body>	→	<content_oneline>
389.	<body>	→	{ <content> }
390.	<content_oneline>	→	<local_declaration>
391.	<content_oneline>	→	<statement>
392.	<content_oneline>	→	<loop_terminator>

393.	<else_clause>	→	else <body>
394.	<else_clause>	→	λ
395.	<push_value>	→	<string_concat>
396.	<push_value>	→	ping(<ping_conversion_value>)
397.	<push_value>	→	λ
398.	<recall_value>	→	<general_expression>
399.	<user_function>	→	spawn <spawn_tail><optional_func>
400.	<user_function>	→	λ
401.	<spawn_tail>	→	<spawn_data_type> Identifier (<parameter>) { <user_body> }
402.	<optional_func>	→	<user_function>
403.	<optional_func>	→	λ
404.	<spawn_data_type>	→	<data_type>
405.	<spawn_data_type>	→	void
406.	<data_type>	→	inter
407.	<data_type>	→	bloat
408.	<data_type>	→	ping
409.	<data_type>	→	pool
410.	<parameter>	→	<data_type> <optional_array> Identifier <additional_param>
411.	<parameter>	→	λ
412.	<optional_array>	→	[<2D_array>]
413.	<optional_array>	→	λ
414.	<2D_array>	→	,
415.	<2D_array>	→	λ
416.	<additional_param>	→	, <data_type><optional_array> Identifier <additional_param>
417.	<additional_param>	→	λ
418.	<user_body>	→	<local_declaration> <user_body>

419.	<user_body>	→	<statement> <user_body>
420.	<user_body>	→	λ

```
dump
comp ping percentage[9] = { "2.8", "14.0", "25.0", "25.0", "18.0", "8.8", "3.3", "1.9", "0.041"};


spawn void base(){
        push("What is your current rank?");


        ping rank = hold();


        if (rank == "iron")
                push("You are in the " + percentage[0] + "of the player base");
        if (rank == "bronze")
                push("You are in the " + percentage[1] + "of the player base");
}
```