CONTEXT FREE GRAMMAR

#	Production	⇒	Production Set
1.	<program>	⇒	<program_definitions> <main_program>
2.	<program_definitions>	⇒	<group_definition> <global_declaration> <function_definition> <program_definitions>
3.	<program_definitions>		λ
4.	<group_definition>	⇒	group identifier { <group_member_list> }
5.	<group_definition>	⇒	λ
6.	<group_member_list>	⇒	<group_member> <group_member_tail>
7.	<group_member_tail>	⇒	<group_member> <group_member_tail>
8.	<group_member_tail>		λ
9.	<group_member>	⇒	<datatype> identifier;
10.	<global_declaration>	⇒	worldwide <datatype> identifier = <expression>;
11.	<global_declaration>	⇒	λ
12.	<type>	⇒	<datatype>
13.	<type>	⇒	identifier
14.	<datatype>	⇒	num
15.	<datatype>	⇒	decimal
16.	<datatype>	⇒	bigdecimal
17.	<datatype>	⇒	letter
18.	<datatype>	⇒	text
19.	<datatype>	⇒	bool
20.	<function_definition>	⇒	define <return_type> identifier (<parameter_list>) { <local_declarations> <statements> <optional_return> }
21.	<function_definition>	⇒	λ
22.	<local_declarations>	⇒	<local_declaration> <local_declarations>

23.	<local_declarations>	⇒	λ
24.	<return_type>	⇒	<datatype>
25.	<return_type>	⇒	empty
26.	<parameter_list>	⇒	<parameter> <parameter_list_tail>
27.	<parameter_list>	⇒	λ
28.	<parameter_list_tail>	⇒	, <parameter> <parameter_list_tail>
29.	<parameter_list_tail>	⇒	λ
30.	<parameter>	⇒	<datatype> identifier
31.	<optional_return>	⇒	<return_statement>
32.	<optional_return>	⇒	λ
33.	<return_statement>	⇒	give <return_tail>
34.	<return_tail>	⇒	<expression>;
35.	<return_tail>	⇒	;
36.	<main_program>	⇒	start { <statements> } finish
37.	<declaration>	⇒	<local_declaration>
38.	<declaration>	⇒	<fixed_declaration>
39.	<declaration>	⇒	<list_declaration>
40.	<local_declaration>	⇒	identifier identifier;
41.	<local_declaration>	⇒	<datatype> identifier = <expression>;
42.	<fixed_declaration>	⇒	fixed <datatype> identifier = <expression>;
43.	<list_declaration>	⇒	list <datatype> identifier = <list_literal_1d>;
44.	<list_declaration>	⇒	list <datatype> identifier = <list_literal_2d>;
45.	<list_literal_1d>	⇒	[<list_elements>]
46.	<list_elements>	⇒	<expression> <list_elements_tail>
47.	<list_elements>	⇒	λ
48.	<list_elements_tail>	⇒	, <expression> <list_elements_tail>

49.	<list_elements_tail>	⇒	λ
50.	<list_literal_2d>	⇒	[<list_rows>]
51.	<list_rows>	⇒	<list_literal_1d> <list_rows_tail>
52.	<list_rows>	⇒	λ
53.	<list_rows_tail>	⇒	, <list_literal_1d> <list_rows_tail>
54.	<list_rows_tail>	⇒	λ
55.	<statements>	⇒	<statement> <statements>
56.	<statements>	⇒	λ
57.	<statement>	⇒	<control_statement>
58.	<statement>	⇒	<assignment_statement>
59.	<statement>	⇒	<function_call_statement>
60.	<statement>	⇒	<declaration>
61.	<statement>	⇒	<io_statement>
62.	<function_call_statement>	⇒	<function_call>
63.	<function_call>	⇒	identifier (<argument_list>)
64.	<assignment_statement>	⇒	<assignable> <assignment_op> <expression> ;
65.	<assignment_statement>	⇒	<assignable> <increment_op> ;
66.	<assignable>	⇒	identifier
67.	<assignable>	⇒	<list_access>
68.	<assignable>	⇒	<group_member_access>
69.	<io_statement>	⇒	show (<argument_list>) ;
70.	<io_statement>	⇒	read(identifier);
71.	<control_statement>	⇒	<check_structure>
72.	<control_statement>	⇒	<select_statement>
73.	<control_statement>	⇒	<iterative_statement>
74.	<check_structure>	⇒	<check_block> <otherwise_chain>

75.	<check_block>	⇒	check (<expression>) { <statements> }
76.	<otherwise_chain>	⇒	<otherwise_check> <otherwise_chain>
77.	<otherwise_chain>	⇒	<optional_otherwise>
78.	<optional_otherwise>	⇒	<otherwise_block>
79.	<optional_otherwise>	⇒	λ
80.	<otherwise_check>	⇒	otherwisecheck (<expression>) { <statements> }
81.	<otherwise_block>	⇒	otherwise { <statements> }
82.	<select_statement>	⇒	select (<expression>) { <option_blocks> <optional_fallback> }
83.	<option_blocks>	⇒	<option_block> <option_blocks>
84.	<option_blocks>	⇒	λ
85.	<option_block>	⇒	option <literal> : <statements> <control_flow> ;
86.	<control_flow>	⇒	stop
87.	<control_flow>	⇒	skip
88.	<optional_fallback>	⇒	<fallback_block>
89.	<optional_fallback>	⇒	λ
90.	<fallback_block>	⇒	fallback : <statements>
91.	<iterative_statement>	⇒	<each_loop>
92.	<iterative_statement>	⇒	<during_loop>
93.	<each_loop>	⇒	each identifier from <expression> to <expression> <step_clause> { <statements> }
94.	<step_clause>	⇒	step <expression>
95.	<step_clause>	⇒	λ
96.	<during_loop>	⇒	during (<expression>) { <statements> }
97.	<expression>	⇒	<logical_or_expression>
98.	<logical_or_expression>	⇒	<logical_and_expression>

99.	<logical_or_expression>	⇒	<logical_or_expression> || <logical_and_expression>
100.	<logical_and_expression>	⇒	<equality_expression>
101.	<logical_and_expression>	⇒	<logical_and_expression> && <equality_expression>
102.	<equality_expression>	⇒	<relational_expression>
103.	<equality_expression>	⇒	<equality_expression> <equality_op> <relational_expression>
104.	<equality_op>	⇒	==
105.	<equality_op>	⇒	!=
106.	<relational_expression>	⇒	<additive_expression>
107.	<relational_expression>	⇒	<relational_expression> <relational_op> <additive_expression>
108.	<relational_op>	⇒	>
109.	<relational_op>	⇒	<
110.	<relational_op>	⇒	>=
111.	<relational_op>	⇒	<=
112.	<additive_expression>	⇒	<multiplicative_expression>
113.	<additive_expression>	⇒	<additive_expression> <add_op> <multiplicative_expression>
114.	<add_op>	⇒	+
115.	<add_op>	⇒	-
116.	<multiplicative_expression>	⇒	<exponentiation_expression>
117.	<multiplicative_expression>	⇒	<multiplicative_expression> <mult_op> <exponentiation_expression>
118.	<mult_op>	⇒	*
119.	<mult_op>	⇒	/
120.	<mult_op>	⇒	%
121.	<exponentiation_expression>	⇒	<unary_expression>

122.	<exponentiation_expression>	⇒	<unary_expression>** <exponentiation_expression>
123.	<unary_expression>	⇒	<unary_op> <postfix_expression>
124.	<unary_expression>	⇒	<postfix_expression>
125.	<unary_op>	⇒	-
126.	<unary_op>	⇒	!
127.	<postfix_expression>	⇒	<primary_expression> <optional_postfix>
128.	<optional_postfix>	⇒	<increment_op>
129.	<optional_postfix>	⇒	λ
130.	<primary_expression>	⇒	(<expression>)
131.	<primary_expression>	⇒	<factor>
132.	<factor>	⇒	<literal>
133.	<factor>	⇒	<variable_reference>
134.	<factor>	⇒	<function_call>
135.	<argument_list>	⇒	<expression> <argument_list_tail>
136.	<argument_list>	⇒	λ
137.	<argument_list_tail>	⇒	, <expression> <argument_list_tail>
138.	<argument_list_tail>	⇒	λ
139.	<variable_reference>	⇒	identifier
140.	<variable_reference>	⇒	<list_access>
141.	<variable_reference>	⇒	<group_member_access>
142.	<list_access>	⇒	identifier [<expression>]
143.	<group_member_access>	⇒	identifier . identifier
144.	<assignment_op>	⇒	=
145.	<assignment_op>	⇒	+=
146.	<assignment_op>	⇒	-=
147.	<assignment_op>	⇒	*=

148.	<assignment_op>	⇒	/=
149.	<assignment_op>	⇒	%=
150.	<assignment_op>	⇒	**=
151.	<increment_op>	⇒	++
152.	<increment_op>	⇒	--
153.	<literal>	⇒	num_lit
154.	<literal>	⇒	decimal_lit
155.	<literal>	⇒	string_lit
156.	<literal>	⇒	char_lit
157.	<literal>	⇒	<bool_literal>
158.	<bool_literal>	⇒	Yes
159.	<bool_literal>	⇒	No
160.	<escape_sequence>	⇒	newline
161.	<escape_sequence>	⇒	tab
162.	<escape_sequence>	⇒	single_quote
163.	<escape_sequence>	⇒	double_quote
164.	<escape_sequence>	⇒	backslash