

XVII. CONTEXT FREE GRAMMAR

CONTEXT FREE GRAMMAR

1	<program>	→	<Prod_comment> <GlobDec> <main> <Prod_comment> <end>
2	<Prod_comment>	→	comment <Prod_comment>
3	<Prod_comment>	→	λ
4	<GlobDec>	→	<datatype> id <Declare1> <Prod_comment>
5	<GlobDec>	→	miss id <functVoid1> <Prod_comment>
6	<GlobDec>	→	struct id <struct1> <Prod_comment>
7	<GlobDec>	→	hold id <const1> <Prod_comment>
8	<GlobDec>	→	λ
9	<Declare1>	→	<DeclareChoice> ; <GlobDec>
10	<Declare1>	→	<functRet> <GlobDec>
11	<functRet1>	→	(<dtypeA>) { <GlobDec> <body> backup (<returnParam>) ; <GlobDec>
12	<struct1>	→	{ <memDec> } ; <GlobDec>
13	<DeclareOption>	→	<datatype> id <Declare> <Prod_comment>
14	<DeclareOption>	→	miss id <functVoid> <Prod_comment>
15	<DeclareOption>	→	struct id <struct> <Prod_comment>
16	<DeclareOption>	→	hold id <const> <Prod_comment>
17	<Declare>	→	<DeclareChoice> ; <DeclareOption>
18	<Declare>	→	<functRet> <DeclareOption>
19	<functRet>	→	(<dtypeA>) { <GlobDec> <body> backup (<returnParam>) ; <DeclareOption>
20	<struct>	→	{ <memDec> } ; <DeclareOption>
21	<const>	→	= <Literal> ; <DeclareOption>
22	<Literal>	→	Numlit
23	<Literal>	→	Declit
24	<Literal>	→	Charlit
25	<Literal>	→	Stringlit
26	<Literal>	→	AFFIRMATIVE
27	<Literal>	→	NEGATIVE

28	<Literal2>	→	Numlit
29	<Literal2>	→	Declit
30	<Literal2>	→	Charlit
31	<Literal2>	→	Stringlit
32	<datatype>	→	unit
33	<datatype>	→	digit
34	<datatype>	→	joe
35	<datatype>	→	company
36	<datatype>	→	response
37	<DeclareChoice>	→	<InitChoice>
38	<DeclareChoice>	→	<N1> <arrayAID>
39	<InitChoice>	→	, id <InitChoice>
40	<InitChoice>	→	= <Literal> <addID>
41	<InitChoice>	→	Λ
42	<addID>	→	, id <InitChoice>
43	<addID>	→	Λ
44	<N1>	→	[<index>] <N2>
45	<N2>	→	[<index>]
46	<N2>	→	Λ
47	<index>	→	Numlit <Smath>
48	<index>	→	id <Smath>
49	<Smath>	→	<operator> <index>
50	<Smath>	→	Λ
51	<arrayAID>	→	= { <ElemChoice> }
52	<arrayAID>	→	Λ
53	<ElemChoice>	→	<Element>
54	<ElemChoice>	→	{ <Element> } <M_Elem>
55	<Element>	→	<Literal2> <addElem>
56	<addElem>	→	, <Element>
57	<addElem>	→	Λ
58	<M_Elem>	→	, { <Element> } <M2_Elem>
59	<M2_Elem>	→	<M_Elem>
60	<M2_Elem>	→	Λ
61	<memDec>	→	<datatype> id <initDec> ; <memDec>
62	<memDec>	→	Λ

63	<initDec>	→	<initDecChoice>
64	<initDec>	→	<N1>
65	<initDec>	→	Λ
66	<initDecChoice>	→	, id <initDecChoice>
67	<initDecChoice>	→	Λ
68	<dtypeA>	→	<datatype> id <ExdtypeA>
69	<dtypeA>	→	Λ
70	<ExdtypeA>	→	, <dtypeA>
71	<ExdtypeA>	→	Λ
72	<returnParam>	→	<Literal>
73	<returnParam>	→	<negate> id <outC>
74	<returnParam>	→	sqrt (returnParam)
75	<returnParam>	→	Λ
76	<negate>	→	\sim
77	<negate>	→	λ
78	<main>	→	PrimaryMission () { <Prod_comment> <body> }
79	<body>	→	<DeclareOption> <body>
80	<body>	→	<print> <body>
81	<body>	→	<scan> <body>
82	<body>	→	<for> <body>
83	<body>	→	<assignChoice> <body>
84	<body>	→	<ifelse> <body>
85	<body>	→	<do_while> <body>
86	<body>	→	<while> <body>
87	<body>	→	<switch> <body>
88	<body>	→	<Prod_comment>
89	<body>	→	Λ
90	<print>	→	post (<postval>) ;
91	<postval>	→	<returnParam> <ConcatLit>
92	<outC>	→	<arrayAID>
93	<outC>	→	(<operand> <addparam>)
94	<outC>	→	. <convert>
95	<ConcatLit>	→	+ <returnParam> <ConcatLit>
96	<ConcatLit>	→	λ

97			
98			
99	<ConcatLit>	→	Λ
100	<ExConcatLit>	→	+ <postval>
101	<ExConcatLit>	→	Λ
102	<scan>	→	capture (<scanVal>) ;
103	<scanVal>	→	# id <addScan>
104	<addScan>	→	, # id <ExtaddScan>
105	<addScan>	→	. id <N1> <ExtaddScan>
106	<addScan>	→	Λ
107	<ExtaddScan>	→	, # id <addScan>
108	<ExtaddScan>	→	Λ
109	<assignChoice>	→	<AccessAssignDtype>
110	<assignChoice>	→	<mmtCond>
111	<assignChoice>	→	<structCall>
112	<assignChoice>	→	swap (<returnParam>) ;
113	<AccessAssignDtype>	→	id <assignValueChoice>
114	<assignValueChoice>	→	<N1> <assignmentInit> ; <assignChoice>
115	<assignValueChoice>	→	= <assignValue> ; <assignChoice>
116	<assignValueChoice>	→	<funcCall> ;
117	<assignValueChoice>	→	. id <structInitial>
118	<assignmentInit>	→	= <assignValue>
119	<assignmentInit>	→	. id = <assignValue>
120	<structInitial>	→	= <initStruct>
121	<structInitial>	→	<AccessValueChoice>
122	<AccessValueChoice>	→	<structMath> <AssignSym> <MathOp>
123	<structCall>	→	struct id <varStruct> ;
124	<varStruct>	→	id <StructArray> <addStructvar>
125	<StructArray>	→	<N1>
126	<StructArray>	→	Λ
127	<addStructvar>	→	, <varStruct>
128	<addStructvar>	→	Λ
129	<funcCall>	→	(<param>)
130	<funcCall>	→	Λ

131	<param>	→	id <addparam>
132	<param>	→	Λ
133	<addparam>	→	, id <addparam>
134	<addparam>	→	Λ
135	<initStruct>	→	<Literal2>
136	<initStruct>	→	id
137	<AssignSym>	→	<oper1> =
138	<oper1>	→	+
139	<oper1>	→	-
140	<oper1>	→	*
141	<oper1>	→	/
142	<oper1>	→	^
143	<oper1>	→	%
144	<oper1>	→	Λ
145	<assignValue>	→	<Literal>
146	<assignValue>	→	id <functCall>
147	<convert>	→	ToJoeRange
148	<convert>	→	Extent
149	<convert>	→	Carry (returnParam) ;
150	<mntCond>	→	<mnt> id
151	<mntCond>	→	id <mnt>
152	<mnt>	→	++
153	<mnt>	→	--
154	<for>	→	inquire (id = <val1> ; <RelOp> ; <mntCond>) { <body> }
155	<val1>	→	Numlit <valPP>
156	<val1>	→	Declit <valPP>
157	<val1>	→	id <valPP>
158	<valPP>	→	<operator> <val1>
159	<valPP>	→	Λ
160	<operator>	→	+
161	<operator>	→	-
162	<operator>	→	*
163	<operator>	→	/
164	<operator>	→	^

165	<operator>	→	%
166	<RelOp>	→	id <RelopExt>
167	<RelopExt>	→	<op1> <Literal> <RelopExt>
168	<RelopExt>	→	Λ
169	<op1>	→	==
170	<op1>	→	>=
171	<op1>	→	<=
172	<op1>	→	!=
173	<LogOper>	→	
174	<LogOper>	→	&
175	<ifelse>	→	inorder (<ifcondition>) { <ifstatement> } <elseif> <else>
176	<ifcondition>	→	<RelOp>
177	<ifcondition>	→	<LogOp>
178	<ifstatement>	→	<body> <break>
179	<ifstatement>	→	backup (<returnParam>) ;
180	<break>	→	abort () ;
181	<LogOp>	→	(<RelOp>) <ExtLogOp>
182	<ExtLogOp>	→	<LogOper> <LogOp>
183	<ExtLogOp>	→	λ
184	<elseif>	→	otherorder (<ifcondition>) { <ifstatement> } <elseif>
185	<elseif>	→	Λ
186	<else>	→	order { <ifstatement> }
187	<else>	→	Λ
188	<do_while>	→	go { <body> } phase (<RelOp>) ;
189	<while>	→	phase (<RelOp>) { <body> }
190	<switch>	→	campaign (id) { <case> <default> }
191	<case>	→	operation <Literal> : <body> <break> <case>
192	<case>	→	Λ
193	<default>	→	action : <body>
194	<default>	→	Λ
195	<MathOp>	→	<operCond> ;
196	<MathOp>	→	Λ

197	<operCond>	→	(<operand> <operExt_s>) <operCondExt>
198	<operCond>	→	<operand> <operExt_s>
199	<operand>	→	<returnParam>
200	<OperationMath>	→	<structMath>
201	<OperationMath>	→	<functCall>
202	<structMath>	→	. id
203	<structMath>	→	Λ
204	<operExt_s>	→	<operator> <operand> <S_MathExt>
205	<operExt_s>	→	(<simMathOp>) <operExt_s>
206	<simMathOp>	→	<operand> <S_MathExt>
207	<S_MathExt>	→	<operator> <operand> <S_MathExt>
208	<S_MathExt>	→	(<simMathOp>) <operExt_s>
209	<S_MathExt>	→	Λ
210	<operCondExt>	→	<operator> <operExt_s>
211	<operCondExt>	→	Λ
212	<end>	→	deploy () ; <Prod_comment>

XVIII. FIRST SET

FIRST SET

	Non-Terminal Symbol	First Set
1	<program>	comment, λ , PrimaryMission, miss, struct, hold, unit, digit, joe, company, response
2	<Prod_comment>	comment, λ
3	<GlobDec>	miss, struct, hold, λ , unit, digit, joe, company, response
4	<Declare1>	;, ,, =, λ , [, (
5	<functRet1>	(
6	<struct1>	{
7	<DeclareOption>	miss, struct, hold, unit, digit, joe, company, response
8	<Declare>	;, ,, =, λ , [, (
9	<functRet>	(
10	<struct>	{
11	<const>	=
12	<Literal>	Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE
13	<Literal2>	Numlit, Declit, Charlit, Stringlit
14	<datatype>	unit, digit, joe, company, response
15	<DeclareChoice>	;, =, λ , [
16	<InitChoice>	;, =, λ
17	<addID>	;, λ
18	<N1>	[
19	<N2>	[, λ
20	<index>	Numlit, id
21	<Smath>	λ , +, -, *, /, ^, %
22	<arrayAID>	=, λ
23	<ElemChoice>	{, Numlit, Declit, Charlit, Stringlit
24	<Element>	Numlit, Declit, Charlit, Stringlit
25	<addElem>	;, λ
26	<M_Elem>	,
27	<M2_Elem>	λ , ,
28	<memDec>	λ , unit, digit, joe, company, response

29	<initDec>	λ , ,, [
30	<initDecChoice>	,, λ
31	<dtypeA>	λ , unit, digit, joe, company, response
32	<ExdtypeA>	,, λ
33	<returnParam>	id, sqrt, λ , Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~
34	<negate>	~, λ
	<main>	PrimaryMission
35	<body>	λ , miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment
36	<print>	post
37	<postval>	id, sqrt, λ , Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, +
38	<outC>	(, ., =, λ
39	<ConcatLit>	+, λ
40	<ExConcatLit>	+, λ
41	<scan>	capture
42	<scanVal>	#
43	<addScan>	,, ., λ
44	<ExtaddScan>	,, λ
45	<assignChoice>	swap, id, struct, ++, --
46	<AccessAssignDtype>	id
	<assignValueChoice>	=, :, ., [, (, λ
47	<assignmentInit>	=, .
48	<structInitial>	=, ., λ , ->, -, *, /, ^, %
49	<AccessValueChoice>	., λ , =, ->, -, *, /, ^, %
50	<structCall>	struct
51	<varStruct>	id
52	<StructArray>	λ , [
53	<addStructvar>	,, λ
54	<functCall>	(, λ
55	<param>	id, λ
56	<addparam>	,, λ

57	<initStruct>	id, Numlit, Declit, Charlit, Stringlit
58	<AssignSym>	=, ->, -, *, /, ^, %, λ
59	<oper1>	->, -, *, /, ^, %, λ
60	<assignValue>	id, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE
61	<convert>	ToJoeRange, Extent, Carry
62	<mntCond>	id, ++, --
63	<mnt>	++, --
64	<for>	inquire
65	<vall>	Numlit, Declit, id
66	<valPP>	λ, +, -, *, /, ^, %
67	<operator>	+, -, *, /, ^, %
68	<RelOp>	id
69	<RelopExt>	λ, ==, >=, <=, !=, <, >
70	<opl>	==, >=, <=, !=, <, >
71	<LogOper>	, &
72	<ifelse>	inorder
73	<ifcondition>	id, (
74	<ifstatement>	backup, λ, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort
75	<break>	abort
76	<LogOp>	(
77	<ExtLogOp>	λ, , &
78	<elseif>	otherorder, λ
79	<else>	order, λ
80	<do_while>	go
81	<while>	phase
82	<switch>	campaign
83	<case>	operation, λ
84	<default>	action, λ
85	<MathOp>	λ, (, id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, +, -, *, /, ^, %

86	<operCond>	(, id, sqrt, λ , Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, \sim , +, -, *, /, \wedge , %
87	<operand>	id, sqrt, λ , Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, \sim
88	<OperationMath>	., λ , (
89	<structMath>	., λ
90	<operExt_s>	(, +, -, *, /, \wedge , %
91	<S_MathExt>	(, λ , +, -, *, /, \wedge , %
92	<operCondExt>	λ , +, -, *, /, \wedge , %
93	<end>	deploy

XIX. FOLLOW SET

FOLLOW SET

Non-Terminal Symbol	Follow Set
<program>	\$
<Prod_comment>), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, PrimaryMission, deploy, abort, }, backup, \$
<GlobDec>), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, PrimaryMission
<Declare1>	comment,), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, PrimaryMission
<struct1>	comment,), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, PrimaryMission
<DeclareOption>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup,), PrimaryMission
<Declare>	comment, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, abort, }, backup,), PrimaryMission

<functRet>	miss, struct, hold, unit, digit, joe, company, response, comment,), capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, PrimaryMission
<struct>	comment, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, abort, }, backup,), PrimaryMission
<const>	comment, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, abort, }, backup,), PrimaryMission
<Literal>	:=, ==, >=, <=, !=, <, >, .., :,), +, (, -, *, /, ^, %
<Literal2>	.. }, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, backup
<datatype>	id
<DeclareChoice>	;
<InitChoice>	;
<addID>	;
<N1>	=, .., .., :,)
<N2>	=, .., .., :,)
<index>]
<Smath>]
<arrayAID>	:=,), +, (, -, *, /, ^, %, ,
<ElemChoice>	}
<Element>	}
<addElem>	}
<M_Elem>	}

<M2_Elem>	}
<memDec>	}
<initDec>	;
<initDecChoice>	;
<dtypeA>)
<ExdtypeA>)
<returnParam>), +, (, -, *, /, ^, %, ,, ;
<negate>	id
<main>	PrimaryMission, comment
<body>	abort, }, backup
<print>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<postval>)
<outC>), +, (, -, *, /, ^, %, ,, ;
<ConcatLit>)
<scan>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<scanVal>)
<addScan>)
<ExtaddScan>)
<assignChoice>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<AccessAssignDtype>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go,

	campaign, inquire, phase, inorder, comment, abort, }, backup
<assignValueChoice>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<assignmentInit>	;
<structInitial>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<AccessValueChoice>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<structCall>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<varStruct>	;
<StructArray>	,, ;
<addStructvar>	;
<funcCall>	;
<param>)
<addparam>)
<initStruct>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup

<AssignSym>	(, id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, +, -, *, /, ^, %, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<oper1>	=
<assignValue>	;
<convert>), +, (, -, *, /, ^, %, ,, ;
<mntCond>), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<mnt>	id,), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<for>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<val1>	;
<valPP>	;
<operator>	(, +, -, *, /, ^, %, id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~,), ;
<RelOp>), ;
<RelopExt>), ;
<op1>	Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE

<LogOp>	(
<ifelse>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<ifcondition>)
<ifstatement>	}
<break>	operation, id, action, }
<LogOp>)
<ExtLogOp>)
<elseif>	order, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<else>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<do_while>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<while>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<switch>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup

<case>	id, action
<default>	}
<MathOp>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
<operCond>	;
<operand>	(, +, -, *, /, ^, %, ,,), ;
<OperationMath>	
<structMath>	=, ->, -, *, /, ^, %
<operExt_s>), ;
<simMathOp>)
<S_MathExt>), ;
<operCondExt>	;
<end>	\$

XX. PREDICT SET

PREDICT SET

#	Expression	Predict
1	$\langle \text{program} \rangle \rightarrow \langle \text{Prod_comment} \rangle$ $\langle \text{GlobDec} \rangle \langle \text{main} \rangle \langle \text{Prod_comment} \rangle$ $\langle \text{end} \rangle$	comment, miss, struct, hold, unit, digit, joe, company, response, PrimaryMission
2	$\langle \text{Prod_comment} \rangle \rightarrow \text{comment}$ $\langle \text{Prod_comment} \rangle$	comment
3	$\langle \text{Prod_comment} \rangle \rightarrow \lambda$), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, PrimaryMission, deploy, abort, }, backup, \$
4	$\langle \text{GlobDec} \rangle \rightarrow \langle \text{datatype} \rangle \text{id}$ $\langle \text{Declare1} \rangle \langle \text{Prod_comment} \rangle$	unit, digit, joe, company, response
5	$\langle \text{GlobDec} \rangle \rightarrow \text{miss id } \langle \text{functVoid1} \rangle$ $\langle \text{Prod_comment} \rangle$	miss
6	$\langle \text{GlobDec} \rangle \rightarrow \text{struct id } \langle \text{struct1} \rangle$ $\langle \text{Prod_comment} \rangle$	struct
7	$\langle \text{GlobDec} \rangle \rightarrow \text{hold id } \langle \text{const1} \rangle$ $\langle \text{Prod_comment} \rangle$	hold
8	$\langle \text{GlobDec} \rangle \rightarrow \lambda$), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, PrimaryMission
9	$\langle \text{Declare1} \rangle \rightarrow \langle \text{DeclareChoice} \rangle ;$ $\langle \text{GlobDec} \rangle$,, =, , ;
10	$\langle \text{Declare1} \rangle \rightarrow \langle \text{functRet} \rangle \langle \text{GlobDec} \rangle$	(
11	$\langle \text{functRet1} \rangle \rightarrow (\langle \text{dtypeA} \rangle) \{$ $\langle \text{GlobDec} \rangle \langle \text{body} \rangle \text{backup } ($ $\langle \text{returnParam} \rangle) ; \langle \text{GlobDec} \rangle$	(
12	$\langle \text{struct1} \rangle \rightarrow \{ \langle \text{memDec} \rangle \} ;$ $\langle \text{GlobDec} \rangle$	{
13	$\langle \text{DeclareOption} \rangle \rightarrow \langle \text{datatype} \rangle \text{id}$ $\langle \text{Declare} \rangle \langle \text{Prod_comment} \rangle$	unit, digit, joe, company, response
14	$\langle \text{DeclareOption} \rangle \rightarrow \text{miss id}$ $\langle \text{functVoid} \rangle \langle \text{Prod_comment} \rangle$	miss
15	$\langle \text{DeclareOption} \rangle \rightarrow \text{struct id}$ $\langle \text{struct} \rangle \langle \text{Prod_comment} \rangle$	struct

16	$\langle \text{DeclareOption} \rangle \rightarrow \text{hold id } \langle \text{const} \rangle$ $\langle \text{Prod_comment} \rangle$	hold
17	$\langle \text{Declare} \rangle \rightarrow \langle \text{DeclareChoice} \rangle ;$ $\langle \text{DeclareOption} \rangle$,, =, , ;
18	$\langle \text{Declare} \rangle \rightarrow \langle \text{functRet} \rangle$ $\langle \text{DeclareOption} \rangle$	(
19	$\langle \text{functRet} \rangle \rightarrow (\langle \text{dtypeA} \rangle) \{$ $\langle \text{GlobDec} \rangle \langle \text{body} \rangle \text{backup } ($ $\langle \text{returnParam} \rangle) ; \langle \text{DeclareOption} \rangle$	(
20	$\langle \text{struct} \rangle \rightarrow \{ \langle \text{memDec} \rangle \} ;$ $\langle \text{DeclareOption} \rangle$	{
21	$\langle \text{const} \rangle \rightarrow = \langle \text{Literal} \rangle ;$ $\langle \text{DeclareOption} \rangle$	=
22	$\langle \text{Literal} \rangle \rightarrow \text{Numlit}$	Numlit
23	$\langle \text{Literal} \rangle \rightarrow \text{Declit}$	Declit
24	$\langle \text{Literal} \rangle \rightarrow \text{Charlit}$	Charlit
25	$\langle \text{Literal} \rangle \rightarrow \text{Stringlit}$	Stringlit
26	$\langle \text{Literal} \rangle \rightarrow \text{AFFIRMATIVE}$	AFFIRMATIVE
27	$\langle \text{Literal} \rangle \rightarrow \text{NEGATIVE}$	NEGATIVE
28	$\langle \text{Literal2} \rangle \rightarrow \text{Numlit}$	Numlit
29	$\langle \text{Literal2} \rangle \rightarrow \text{Declit}$	Declit
30	$\langle \text{Literal2} \rangle \rightarrow \text{Charlit}$	Charlit
31	$\langle \text{Literal2} \rangle \rightarrow \text{Stringlit}$	Stringlit
32	$\langle \text{datatype} \rangle \rightarrow \text{unit}$	unit
33	$\langle \text{datatype} \rangle \rightarrow \text{digit}$	digit
34	$\langle \text{datatype} \rangle \rightarrow \text{joe}$	joe
35	$\langle \text{datatype} \rangle \rightarrow \text{company}$	company
36	$\langle \text{datatype} \rangle \rightarrow \text{response}$	response
37	$\langle \text{DeclareChoice} \rangle \rightarrow \langle \text{InitChoice} \rangle$,, =
38	$\langle \text{DeclareChoice} \rangle \rightarrow \langle \text{N1} \rangle \langle \text{arrayAID} \rangle$	
39	$\langle \text{InitChoice} \rangle \rightarrow , \text{id } \langle \text{InitChoice} \rangle$,
40	$\langle \text{InitChoice} \rangle \rightarrow = \langle \text{Literal} \rangle$ $\langle \text{addID} \rangle$	=
41	$\langle \text{InitChoice} \rangle \rightarrow \lambda$;
42	$\langle \text{addID} \rangle \rightarrow , \text{id } \langle \text{InitChoice} \rangle$,
43	$\langle \text{addID} \rangle \rightarrow \lambda$;
44	$\langle \text{N1} \rangle \rightarrow [\langle \text{index} \rangle] \langle \text{N2} \rangle$	
45	$\langle \text{N2} \rangle \rightarrow [\langle \text{index} \rangle]$	
46	$\langle \text{N2} \rangle \rightarrow \lambda$	=, ., ., ., ;,)
47	$\langle \text{index} \rangle \rightarrow \text{Numlit } \langle \text{Smath} \rangle$	Numlit
48	$\langle \text{index} \rangle \rightarrow \text{id } \langle \text{Smath} \rangle$	id

49	$\langle \text{Smath} \rangle \rightarrow \langle \text{operator} \rangle \langle \text{index} \rangle$	+, -, *, /, ^, %
50	$\langle \text{Smath} \rangle \rightarrow \lambda$	
51	$\langle \text{arrayAID} \rangle \rightarrow = \{ \langle \text{ElemChoice} \rangle \}$	=
52	$\langle \text{arrayAID} \rangle \rightarrow \lambda$;,), +, (, -, *, /, ^, %, ,
53	$\langle \text{ElemChoice} \rangle \rightarrow \langle \text{Element} \rangle$	Numlit, Declit, Charlit, Stringlit
54	$\langle \text{ElemChoice} \rangle \rightarrow \{ \langle \text{Element} \rangle \}$ $\langle \text{M_Elem} \rangle$	{
55	$\langle \text{Element} \rangle \rightarrow \langle \text{Literal2} \rangle \langle \text{addElem} \rangle$	Numlit, Declit, Charlit, Stringlit
56	$\langle \text{addElem} \rangle \rightarrow , \langle \text{Element} \rangle$,
57	$\langle \text{addElem} \rangle \rightarrow \lambda$	}
58	$\langle \text{M_Elem} \rangle \rightarrow , \{ \langle \text{Element} \rangle \}$ $\langle \text{M2_Elem} \rangle$,
59	$\langle \text{M2_Elem} \rangle \rightarrow \langle \text{M_Elem} \rangle$,
60	$\langle \text{M2_Elem} \rangle \rightarrow \lambda$	}
61	$\langle \text{memDec} \rangle \rightarrow \langle \text{datatype} \rangle \text{id}$ $\langle \text{initDec} \rangle ; \langle \text{memDec} \rangle$	unit, digit, joe, company, response
62	$\langle \text{memDec} \rangle \rightarrow \lambda$	}
63	$\langle \text{initDec} \rangle \rightarrow \langle \text{initDecChoice} \rangle$,
64	$\langle \text{initDec} \rangle \rightarrow \langle \text{N1} \rangle$	
65	$\langle \text{initDec} \rangle \rightarrow \lambda$;
66	$\langle \text{initDecChoice} \rangle \rightarrow , \text{id}$ $\langle \text{initDecChoice} \rangle$,
67	$\langle \text{initDecChoice} \rangle \rightarrow \lambda$;
68	$\langle \text{dtypeA} \rangle \rightarrow \langle \text{datatype} \rangle \text{id}$ $\langle \text{ExdtypeA} \rangle$	unit, digit, joe, company, response
69	$\langle \text{dtypeA} \rangle \rightarrow \lambda$)
70	$\langle \text{ExdtypeA} \rangle \rightarrow , \langle \text{dtypeA} \rangle$,
71	$\langle \text{ExdtypeA} \rangle \rightarrow \lambda$)
72	$\langle \text{returnParam} \rangle \rightarrow \langle \text{Literal} \rangle$	Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE
73	$\langle \text{returnParam} \rangle \rightarrow \langle \text{negate} \rangle \text{id}$ $\langle \text{outC} \rangle$	~, id
74	$\langle \text{returnParam} \rangle \rightarrow \text{sqrt} ($ $\text{returnParam})$	sqrt
75	$\langle \text{returnParam} \rangle \rightarrow \lambda$), +, (, -, *, /, ^, %, ,, ;
76	$\langle \text{negate} \rangle \rightarrow \sim$	~
77	$\langle \text{negate} \rangle \rightarrow \lambda$	id
78	$\langle \text{main} \rangle \rightarrow \text{PrimaryMission} () \{$ $\langle \text{Prod_comment} \rangle \langle \text{body} \rangle \}$	PrimaryMission
79	$\langle \text{body} \rangle \rightarrow \langle \text{DeclareOption} \rangle \langle \text{body} \rangle$	miss, struct, hold, unit, digit, joe, company, response

80	$\langle \text{body} \rangle \rightarrow \langle \text{print} \rangle \langle \text{body} \rangle$	post
81	$\langle \text{body} \rangle \rightarrow \langle \text{scan} \rangle \langle \text{body} \rangle$	capture
82	$\langle \text{body} \rangle \rightarrow \langle \text{for} \rangle \langle \text{body} \rangle$	inquire
83	$\langle \text{body} \rangle \rightarrow \langle \text{assignChoice} \rangle \langle \text{body} \rangle$	swap, id, struct, ++, --
84	$\langle \text{body} \rangle \rightarrow \langle \text{ifelse} \rangle \langle \text{body} \rangle$	inorder
85	$\langle \text{body} \rangle \rightarrow \langle \text{do_while} \rangle \langle \text{body} \rangle$	go
86	$\langle \text{body} \rangle \rightarrow \langle \text{while} \rangle \langle \text{body} \rangle$	phase
87	$\langle \text{body} \rangle \rightarrow \langle \text{switch} \rangle \langle \text{body} \rangle$	campaign
88	$\langle \text{body} \rangle \rightarrow \langle \text{Prod_comment} \rangle$	comment
89	$\langle \text{body} \rangle \rightarrow \lambda$	abort, }, backup
90	$\langle \text{print} \rangle \rightarrow \text{post} (\langle \text{postval} \rangle) ;$	post
91	$\langle \text{postval} \rangle \rightarrow \langle \text{returnParam} \rangle$ $\langle \text{ConcatLit} \rangle$	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, +
92	$\langle \text{outC} \rangle \rightarrow \langle \text{arrayAID} \rangle$	=
93	$\langle \text{outC} \rangle \rightarrow (\langle \text{operand} \rangle \langle \text{addparam} \rangle)$	(
94	$\langle \text{outC} \rangle \rightarrow . \langle \text{convert} \rangle$.
95	$\langle \text{ConcatLit} \rangle \rightarrow + \langle \text{returnParam} \rangle$ $\langle \text{ConcatLit} \rangle$	+
96	$\langle \text{ConcatLit} \rangle \rightarrow \lambda$)
97	$\langle \text{ExConcatLit} \rangle \rightarrow + \langle \text{postval} \rangle$	+
98	$\langle \text{ExConcatLit} \rangle \rightarrow \lambda$	
99	$\langle \text{scan} \rangle \rightarrow \text{capture} (\langle \text{scanVal} \rangle) ;$	capture
100	$\langle \text{scanVal} \rangle \rightarrow \# \text{id} \langle \text{addScan} \rangle$	#
101	$\langle \text{addScan} \rangle \rightarrow , \# \text{id} \langle \text{ExtaddScan} \rangle$,
102	$\langle \text{addScan} \rangle \rightarrow . \text{id} \langle \text{N1} \rangle$ $\langle \text{ExtaddScan} \rangle$.
103	$\langle \text{addScan} \rangle \rightarrow \lambda$)
104	$\langle \text{ExtaddScan} \rangle \rightarrow , \# \text{id} \langle \text{addScan} \rangle$,
105	$\langle \text{ExtaddScan} \rangle \rightarrow \lambda$)
106	$\langle \text{assignChoice} \rangle \rightarrow$ $\langle \text{AccessAssignDtype} \rangle$	id
107	$\langle \text{assignChoice} \rangle \rightarrow \langle \text{mntCond} \rangle$	id, ++, --
108	$\langle \text{assignChoice} \rangle \rightarrow \langle \text{structCall} \rangle$	struct

10 9	$\langle \text{assignChoice} \rangle \rightarrow \text{swap} ($ $\langle \text{returnParam} \rangle) ;$	swap
11 0	$\langle \text{AccessAssignDtype} \rangle \rightarrow \text{id}$ $\langle \text{assignValueChoice} \rangle$	id
11 1	$\langle \text{assignValueChoice} \rangle \rightarrow \langle \text{N1} \rangle$ $\langle \text{assignmentInit} \rangle ; \langle \text{assignChoice} \rangle$	
11 2	$\langle \text{assignValueChoice} \rangle \rightarrow =$ $\langle \text{assignValue} \rangle ; \langle \text{assignChoice} \rangle$	=
11 3	$\langle \text{assignValueChoice} \rangle \rightarrow \langle \text{functCall} \rangle$;	(, ;
11 4	$\langle \text{assignValueChoice} \rangle \rightarrow . \text{id}$ $\langle \text{structInitial} \rangle$.
11 5	$\langle \text{assignmentInit} \rangle \rightarrow =$ $\langle \text{assignValue} \rangle$	=
11 6	$\langle \text{assignmentInit} \rangle \rightarrow . \text{id} =$ $\langle \text{assignValue} \rangle$.
11 7	$\langle \text{structInitial} \rangle \rightarrow = \langle \text{initStruct} \rangle$	=
11 8	$\langle \text{structInitial} \rangle \rightarrow$ $\langle \text{AccessValueChoice} \rangle$., =, ->, -, *, /, ^, %
11 9	$\langle \text{AccessValueChoice} \rangle \rightarrow$ $\langle \text{structMath} \rangle \langle \text{AssignSym} \rangle \langle \text{MathOp} \rangle$., =, ->, -, *, /, ^, %
12 0	$\langle \text{structCall} \rangle \rightarrow \text{struct id}$ $\langle \text{varStruct} \rangle ;$	struct
12 1	$\langle \text{varStruct} \rangle \rightarrow \text{id} \langle \text{StructArray} \rangle$ $\langle \text{addStructvar} \rangle$	id
12 2	$\langle \text{StructArray} \rangle \rightarrow \langle \text{N1} \rangle$	
12 3	$\langle \text{StructArray} \rangle \rightarrow \lambda$., ;
12 4	$\langle \text{addStructvar} \rangle \rightarrow , \langle \text{varStruct} \rangle$,
12 5	$\langle \text{addStructvar} \rangle \rightarrow \lambda$;
12 6	$\langle \text{functCall} \rangle \rightarrow (\langle \text{param} \rangle)$	(
12 7	$\langle \text{functCall} \rangle \rightarrow \lambda$;
12 8	$\langle \text{param} \rangle \rightarrow \text{id} \langle \text{addparam} \rangle$	id

12 9	$\langle \text{param} \rangle \rightarrow \lambda$)
13 0	$\langle \text{addparam} \rangle \rightarrow , \text{id } \langle \text{addparam} \rangle$,
13 1	$\langle \text{addparam} \rangle \rightarrow \lambda$)
13 2	$\langle \text{initStruct} \rangle \rightarrow \langle \text{Literal2} \rangle$	Numlit, Declit, Charlit, Stringlit
13 3	$\langle \text{initStruct} \rangle \rightarrow \text{id}$	id
13 4	$\langle \text{AssignSym} \rangle \rightarrow \langle \text{oper1} \rangle =$	->, -, *, /, ^, %, =
13 5	$\langle \text{oper1} \rangle \rightarrow \rightarrow +$	->
13 6	$\langle \text{oper1} \rangle \rightarrow -$	-
13 7	$\langle \text{oper1} \rangle \rightarrow *$	*
13 8	$\langle \text{oper1} \rangle \rightarrow /$	/
13 9	$\langle \text{oper1} \rangle \rightarrow ^$	^
14 0	$\langle \text{oper1} \rangle \rightarrow \%$	%
14 1	$\langle \text{oper1} \rangle \rightarrow \lambda$	=
14 2	$\langle \text{assignValue} \rangle \rightarrow \langle \text{Literal} \rangle$	Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE
14 3	$\langle \text{assignValue} \rangle \rightarrow \text{id } \langle \text{functCall} \rangle$	id
14 4	$\langle \text{convert} \rangle \rightarrow \text{ToJoeRange}$	ToJoeRange
14 5	$\langle \text{convert} \rangle \rightarrow \text{Extent}$	Extent
14 6	$\langle \text{convert} \rangle \rightarrow \text{Carry (returnParam) ;}$	Carry
14 7	$\langle \text{mntCond} \rangle \rightarrow \langle \text{mnt} \rangle \text{id}$	++, --
14 8	$\langle \text{mntCond} \rangle \rightarrow \text{id } \langle \text{mnt} \rangle$	id

14 9	$\langle \text{mnt} \rangle \rightarrow ++$	++
15 0	$\langle \text{mnt} \rangle \rightarrow --$	--
15 1	$\langle \text{for} \rangle \rightarrow \text{inquire (id = } \langle \text{val1} \rangle ; \langle \text{RelOp} \rangle ; \langle \text{mntCond} \rangle) \{ \langle \text{body} \rangle \}$	inquire
15 2	$\langle \text{val1} \rangle \rightarrow \text{Numlit } \langle \text{valPP} \rangle$	Numlit
15 3	$\langle \text{val1} \rangle \rightarrow \text{Declit } \langle \text{valPP} \rangle$	Declit
15 4	$\langle \text{val1} \rangle \rightarrow \text{id } \langle \text{valPP} \rangle$	id
15 5	$\langle \text{valPP} \rangle \rightarrow \langle \text{operator} \rangle \langle \text{val1} \rangle$	+, -, *, /, ^, %
15 6	$\langle \text{valPP} \rangle \rightarrow \lambda$;
15 7	$\langle \text{operator} \rangle \rightarrow +$	+
15 8	$\langle \text{operator} \rangle \rightarrow -$	-
15 9	$\langle \text{operator} \rangle \rightarrow *$	*
16 0	$\langle \text{operator} \rangle \rightarrow /$	/
16 1	$\langle \text{operator} \rangle \rightarrow ^$	^
16 2	$\langle \text{operator} \rangle \rightarrow \%$	%
16 3	$\langle \text{RelOp} \rangle \rightarrow \text{id } \langle \text{RelOpExt} \rangle$	id
16 4	$\langle \text{RelOpExt} \rangle \rightarrow \langle \text{op1} \rangle \langle \text{Literal} \rangle \langle \text{RelOpExt} \rangle$	==, >=, <=, !=, <, >
16 5	$\langle \text{RelOpExt} \rangle \rightarrow \lambda$), ;
16 6	$\langle \text{op1} \rangle \rightarrow ==$	==
16 7	$\langle \text{op1} \rangle \rightarrow >=$	>=
16 8	$\langle \text{op1} \rangle \rightarrow <=$	<=

16 9	$\langle \text{opl} \rangle \rightarrow !=$!=
17 0	$\langle \text{opl} \rangle \rightarrow <$	<
17 1	$\langle \text{opl} \rangle \rightarrow >$	>
17 2	$\langle \text{LogOper} \rangle \rightarrow \text{oror}$	oror
17 3	$\langle \text{LogOper} \rangle \rightarrow \&$	&
17 4	$\langle \text{ifelse} \rangle \rightarrow \text{inorder (}$ $\langle \text{ifcondition} \rangle) \{ \langle \text{ifstatement} \rangle \}$ $\langle \text{elseif} \rangle \langle \text{else} \rangle$	inorder
17 5	$\langle \text{ifcondition} \rangle \rightarrow \langle \text{RelOp} \rangle$	id
17 6	$\langle \text{ifcondition} \rangle \rightarrow \langle \text{LogOp} \rangle$	(
17 7	$\langle \text{ifstatement} \rangle \rightarrow \langle \text{body} \rangle \langle \text{break} \rangle$	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort
17 8	$\langle \text{ifstatement} \rangle \rightarrow \text{backup (}$ $\langle \text{returnParam} \rangle) ;$	backup
17 9	$\langle \text{ifstatement} \rangle \rightarrow \lambda$	}
18 0	$\langle \text{break} \rangle \rightarrow \text{abort () ;}$	abort
18 1	$\langle \text{LogOp} \rangle \rightarrow (\langle \text{RelOp} \rangle) \langle \text{ExtLogOp} \rangle$	(
18 2	$\langle \text{ExtLogOp} \rangle \rightarrow \langle \text{LogOper} \rangle \langle \text{LogOp} \rangle$	oror, &
18 3	$\langle \text{ExtLogOp} \rangle \rightarrow \lambda$)
18 4	$\langle \text{elseif} \rangle \rightarrow \text{otherorder (}$ $\langle \text{ifcondition} \rangle) \{ \langle \text{ifstatement} \rangle \}$ $\langle \text{elseif} \rangle$	otherorder
18 5	$\langle \text{elseif} \rangle \rightarrow \lambda$	order, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go,

		campaign, inquire, phase, inorder, comment, abort, }, backup
18 6	$\langle \text{else} \rangle \rightarrow \text{order } \{ \langle \text{ifstatement} \rangle \}$	order
18 7	$\langle \text{else} \rangle \rightarrow \lambda$	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
18 8	$\langle \text{do_while} \rangle \rightarrow \text{go } \{ \langle \text{body} \rangle \} \text{ phase } (\langle \text{RelOp} \rangle) ;$	go
18 9	$\langle \text{while} \rangle \rightarrow \text{phase } (\langle \text{RelOp} \rangle) \{ \langle \text{body} \rangle \}$	phase
19 0	$\langle \text{switch} \rangle \rightarrow \text{campaign } (\text{id}) \{ \langle \text{case} \rangle \langle \text{default} \rangle \}$	campaign
19 1	$\langle \text{case} \rangle \rightarrow \text{operation } \langle \text{Literal} \rangle : \langle \text{body} \rangle \langle \text{break} \rangle \langle \text{case} \rangle$	operation
19 2	$\langle \text{case} \rangle \rightarrow \lambda$	id, action
19 3	$\langle \text{default} \rangle \rightarrow \text{action} : \langle \text{body} \rangle$	action
19 4	$\langle \text{default} \rangle \rightarrow \lambda$	}
19 5	$\langle \text{MathOp} \rangle \rightarrow \langle \text{operCond} \rangle ;$	(, id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, +, -, *, /, ^, %
19 6	$\langle \text{MathOp} \rangle \rightarrow \lambda$	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
19 7	$\langle \text{operCond} \rangle \rightarrow (\langle \text{operand} \rangle \langle \text{operExt}_s \rangle) \langle \text{operCondExt} \rangle$	(
19 8	$\langle \text{operCond} \rangle \rightarrow \langle \text{operand} \rangle \langle \text{operExt}_s \rangle$	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, (, +, -, *, /, ^, %
19 9	$\langle \text{operand} \rangle \rightarrow \langle \text{returnParam} \rangle$	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~
20 0	$\langle \text{OperationMath} \rangle \rightarrow \langle \text{structMath} \rangle$.

20 1	$\langle \text{OperationMath} \rangle \rightarrow \langle \text{functCall} \rangle$	(
20 2	$\langle \text{structMath} \rangle \rightarrow . \text{ id}$.
20 3	$\langle \text{structMath} \rangle \rightarrow \lambda$	=, -, -, *, /, ^, %
20 4	$\langle \text{operExt_s} \rangle \rightarrow \langle \text{operator} \rangle$ $\langle \text{operand} \rangle \langle \text{S_MathExt} \rangle$	+, -, *, /, ^, %
20 5	$\langle \text{operExt_s} \rangle \rightarrow (\langle \text{simMathOp} \rangle)$ $\langle \text{operExt_s} \rangle$	(
20 6	$\langle \text{simMathOp} \rangle \rightarrow \langle \text{operand} \rangle$ $\langle \text{S_MathExt} \rangle$	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, (, +, -, *, /, ^, %
20 7	$\langle \text{S_MathExt} \rangle \rightarrow \langle \text{operator} \rangle$ $\langle \text{operand} \rangle \langle \text{S_MathExt} \rangle$	+, -, *, /, ^, %
20 8	$\langle \text{S_MathExt} \rangle \rightarrow (\langle \text{simMathOp} \rangle)$ $\langle \text{operExt_s} \rangle$	(
20 9	$\langle \text{S_MathExt} \rangle \rightarrow \lambda$), ;
21 0	$\langle \text{operCondExt} \rangle \rightarrow \langle \text{operator} \rangle$ $\langle \text{operExt_s} \rangle$	+, -, *, /, ^, %
21 1	$\langle \text{operCondExt} \rangle \rightarrow \lambda$;
21 2	$\langle \text{end} \rangle \rightarrow \text{deploy} () ;$ $\langle \text{Prod_comment} \rangle$	deploy

XXI. PREDICT TABLE