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>	→	~
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>	→	<mntCond>
111	<assignChoice>	→	<structCall>
112	<assignChoice>	→	swap (<returnParam>) ;
113	<AccessAssignDtype>	→	id <assignValueChoice>
114	<assignValueChoice>	→	<N1> <assignmentInit> ; <assignChoice>
115	<assignValueChoice>	→	= <assignValue> ; <assignChoice>
116	<assignValueChoice>	→	<functCall> ;
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	<functCall>	→	(<param>)
130	<functCall>	→	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	<val1>	Numlit, Declit, id
66	<valPP>	λ, +, -, *, /, ^, %
67	<operator>	+, -, *, /, ^, %
68	<RelOp>	id
69	<RelopExt>	λ, ==, >=, <=, !=, <, >
70	<op1>	==, >=, <=, !=, <, >
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, ~, +, -, *, /, ^, %
87	<operand>	id, sqrt, λ, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~
88	<OperationMath>	., λ, (
89	<structMath>	., λ
90	<operExt_s>	(, +, -, *, /, ^, %
91	<S_MathExt>	(, λ, +, -, *, /, ^, %
92	<operCondExt>	λ, +, -, *, /, ^, %
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>	;
<functCall>	;
<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

<LogOper>	(
<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	<program> → <Prod_comment> <GlobDec> <main> <Prod_comment> <end>	comment, miss, struct, hold, unit, digit, joe, company, response, PrimaryMission
2	<Prod_comment> → comment <Prod_comment>	comment
3	<Prod_comment> → λ	), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, PrimaryMission, deploy, abort, }, backup, $
4	<GlobDec> → <datatype> id <Declare1> <Prod_comment>	unit, digit, joe, company, response
5	<GlobDec> → miss id <functVoid1> <Prod_comment>	miss
6	<GlobDec> → struct id <struct1> <Prod_comment>	struct
7	<GlobDec> → hold id <const1> <Prod_comment>	hold
8	<GlobDec> → λ	), miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, PrimaryMission
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>	unit, digit, joe, company, response
14	<DeclareOption> → miss id <functVoid> <Prod_comment>	miss
15	<DeclareOption> → struct id <struct> <Prod_comment>	struct

16	<DeclareOption> → hold id <const> <Prod_comment>	hold
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	Numlit
23	<Literal> → Declit	Declit
24	<Literal> → Charlit	Charlit
25	<Literal> → Stringlit	Stringlit
26	<Literal> → AFFIRMATIVE	AFFIRMATIVE
27	<Literal> → NEGATIVE	NEGATIVE
28	<Literal2> → Numlit	Numlit
29	<Literal2> → Declit	Declit
30	<Literal2> → Charlit	Charlit
31	<Literal2> → Stringlit	Stringlit
32	<datatype> → unit	unit
33	<datatype> → digit	digit
34	<datatype> → joe	joe
35	<datatype> → company	company
36	<datatype> → response	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>	Numlit
48	<index> → id <Smath>	id

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

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

109	<assignChoice> → swap (<returnParam>) ;	swap
110	<AccessAssignDtype> → id <assignValueChoice>	id
111	<assignValueChoice> → <N1> <assignmentInit> ; <assignChoice>	
112	<assignValueChoice> → = <assignValue> ; <assignChoice>	=
113	<assignValueChoice> → <functCall> ;	(, ;
114	<assignValueChoice> → . id <structInitial>	.
115	<assignmentInit> → = <assignValue>	=
116	<assignmentInit> → . id = <assignValue>	.
117	<structInitial> → = <initStruct>	=
118	<structInitial> → <AccessValueChoice>	., =, ->, -, *, /, ^, %
119	<AccessValueChoice> → <structMath> <AssignSym> <MathOp>	., =, ->, -, *, /, ^, %
120	<structCall> → struct id <varStruct> ;	struct
121	<varStruct> → id <StructArray> <addStructvar>	id
122	<StructArray> → <N1>	
123	<StructArray> → λ	,, ;
124	<addStructvar> → , <varStruct>	,
125	<addStructvar> → λ	;
126	<functCall> → (<param>)	(
127	<functCall> → λ	;
128	<param> → id <addparam>	id

129	<param> → λ	)
130	<addparam> → , id <addparam>	,
131	<addparam> → λ	)
132	<initStruct> → <Literal2>	Numlit, Declit, Charlit, Stringlit
133	<initStruct> → id	id
134	<AssignSym> → <oper1> =	->, -, *, /, ^, %, =
135	<oper1> → → +	->
136	<oper1> → -	-
137	<oper1> → *	*
138	<oper1> → /	/
139	<oper1> → ^	^
140	<oper1> → %	%
141	<oper1> → λ	=
142	<assignValue> → <Literal>	Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE
143	<assignValue> → id <functCall>	id
144	<convert> → ToJoeRange	ToJoeRange
145	<convert> → Extent	Extent
146	<convert> → Carry (returnParam) ;	Carry
147	<mntCond> → <mnt> id	++, --
148	<mntCond> → id <mnt>	id

149	<mnt> → ++	++
150	<mnt> → --	--
151	<for> → inquire (id = <val1> ; <RelOp> ; <mntCond>) { <body> }	inquire
152	<val1> → Numlit <valPP>	Numlit
153	<val1> → Declit <valPP>	Declit
154	<val1> → id <valPP>	id
155	<valPP> → <operator> <val1>	+, -, *, /, ^, %
156	<valPP> → λ	;
157	<operator> → +	+
158	<operator> → -	-
159	<operator> → *	*
160	<operator> → /	/
161	<operator> → ^	^
162	<operator> → %	%
163	<RelOp> → id <RelopExt>	id
164	<RelopExt> → <op1> <Literal> <RelopExt>	==, >=, <=, !=, <, >
165	<RelopExt> → λ	), ;
166	<op1> → ==	==
167	<op1> → >=	>=
168	<op1> → <=	<=

169	<op1> → !=	!=
170	<op1> → <	<
171	<op1> → >	>
172	<LogOper> → oror	oror
173	<LogOper> → &	&
174	<ifelse> → inorder (<ifcondition>) { <ifstatement> } <elseif> <else>	inorder
175	<ifcondition> → <RelOp>	id
176	<ifcondition> → <LogOp>	(
177	<ifstatement> → <body> <break>	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort
178	<ifstatement> → backup (<returnParam>) ;	backup
179	<ifstatement> → λ	}
180	<break> → abort ( ) ;	abort
181	<LogOp> → (<RelOp>) <ExtLogOp>	(
182	<ExtLogOp> → <LogOper> <LogOp>	oror, &
183	<ExtLogOp> → λ	)
184	<elseif> → otherorder (<ifcondition>) { <ifstatement> } <elseif>	otherorder
185	<elseif> → λ	order, miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go,

		campaign, inquire, phase, inorder, comment, abort, }, backup
186	<else> → order { <ifstatement> }	order
187	<else> → λ	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
188	<do_while> → go { <body> } phase (<RelOp>) ;	go
189	<while> → phase (<RelOp>) { <body> }	phase
190	<switch> → campaign (id) { <case> <default> }	campaign
191	<case> → operation <Literal> : <body> <break> <case>	operation
192	<case> → λ	id, action
193	<default> → action : <body>	action
194	<default> → λ	}
195	<MathOp> → <operCond> ;	(, id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, +, -, *, /, ^, %
196	<MathOp> → λ	miss, struct, hold, unit, digit, joe, company, response, capture, post, swap, id, ++, --, go, campaign, inquire, phase, inorder, comment, abort, }, backup
197	<operCond> → (<operand> <operExt_s>) <operCondExt>	(
198	<operCond> → <operand> <operExt_s>	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, (, +, -, *, /, ^, %
199	<operand> → <returnParam>	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~
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>	id, sqrt, Numlit, Declit, Charlit, Stringlit, AFFIRMATIVE, NEGATIVE, ~, (, +, -, *, /, ^, %
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>	deploy

XXI. PREDICT TABLE