

```
1 Token: separator Lexeme: $$
2 <Rat16F> -> $$ <Opt Function Definitions>
3 $$ <Opt Declaration List> <Statement List> $$
4
5 Token: keyword Lexeme: function
6 <Opt Function Definitions> -> <Function Definitions> | <Empty>
7 <Function Definitions> -> <Function> | <Function> <Function Definitions>
8 <Function> -> function <Identifier> [ <Opt Parameter List> ] <Opt Declaration List> <Body>
9
10 Token: identifier Lexeme: Hello
11
12 Token: separator Lexeme: [
13
14 Token: identifier Lexeme: me
15 <Opt Parameter List> -> <Parameter List> | <Empty>
16 <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
17 <Parameter> -> <IDs> : <Qualifier>
18 <IDs> -> <Identifier> | <Identifier>, <IDs>
19
20 Token: separator Lexeme: :
21
22 Token: keyword Lexeme: real
23 <Qualifier> -> integer | boolean | real
24
25 Token: separator Lexeme: ,
26
27 Token: identifier Lexeme: alice
28 <Parameter> -> <IDs> : <Qualifier>
29 <IDs> -> <Identifier> | <Identifier>, <IDs>
30
31 Token: separator Lexeme: :
32
33 Token: keyword Lexeme: integer
34 <Qualifier> -> integer | boolean | real
35
36 Token: separator Lexeme: ,
37
38 Token: identifier Lexeme: bob
39 <Parameter> -> <IDs> : <Qualifier>
40 <IDs> -> <Identifier> | <Identifier>, <IDs>
41
42 Token: separator Lexeme: :
43
44 Token: keyword Lexeme: boolean
45 <Qualifier> -> integer | boolean | real
46
47 Token: separator Lexeme: ]
48
49 Token: keyword Lexeme: boolean
50 <Opt Declaration List> -> <Declaration List> | <Empty>
51 <Declaration List> -> <Declaration> ; | <Declaration> ; <Declaration List>
52 <Declaration> -> <Qualifier> <IDs>
53 <Qualifier> -> integer | boolean | real
54
55 Token: identifier Lexeme: modest
56 <IDs> -> <Identifier> | <Identifier>, <IDs>
57
58 Token: separator Lexeme: ;
59
60 Token: keyword Lexeme: boolean
61 <Declaration List> -> <Declaration> ; | <Declaration> ; <Declaration List>
62 <Declaration> -> <Qualifier> <IDs>
63 <Qualifier> -> integer | boolean | real
64
65 Token: identifier Lexeme: mouse
66 <IDs> -> <Identifier> | <Identifier>, <IDs>
67
```

```
68 Token: separator Lexeme: ;
69
70 Token: keyword Lexeme: real
71 <Declaration List> -> <Declaration> ; | <Declaration> ; <Declaration List>
72 <Declaration> -> <Qualifier> <IDs>
73 <Qualifier> -> integer | boolean | real
74
75 Token: identifier Lexeme: pink
76 <IDs> -> <Identifier> | <Identifier>, <IDs>
77
78 Token: separator Lexeme: ;
79
80 Token: keyword Lexeme: integer
81 <Declaration List> -> <Declaration> ; | <Declaration> ; <Declaration List>
82 <Declaration> -> <Qualifier> <IDs>
83 <Qualifier> -> integer | boolean | real
84
85 Token: identifier Lexeme: floyd
86 <IDs> -> <Identifier> | <Identifier>, <IDs>
87
88 Token: separator Lexeme: ;
89
90 Token: separator Lexeme: {
91 <Declaration List> -> <Declaration> ; | <Declaration> ; <Declaration List>
92 <Declaration> -> <Qualifier> <IDs>
93 <Qualifier> -> integer | boolean | real
94 <Body> -> { <Statement List> }
95
96 Token: identifier Lexeme: alice
97 <Statement List> -> <Statement> | <Statement> <Statement List>
98 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
99 <Compound> -> { <Statement List> }
100 <Assign> -> <Identifier> := <Expression>;
101
102 Token: operator Lexeme: :=
103
104 Token: identifier Lexeme: me
105 <Expression> -> <Term> <Expression Prime>
106 <Term> -> <Factor> <Term Prime>
107 <Factor> -> - <Primary> | <Primary>
108 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
false
109
110 Token: separator Lexeme: ;
111 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
112 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
113
114 Token: keyword Lexeme: if
115 <If> -> if (<Condition>) <Statement> endif |
116 if (<Condition>) <Statement> else <Statement> endif
117
118 Token: separator Lexeme: (
119
120 Token: identifier Lexeme: pink
121 <Condition> -> <Expression> <Relop> <Expression>
122 <Expression> -> <Term> <Expression Prime>
123 <Term> -> <Factor> <Term Prime>
124 <Factor> -> - <Primary> | <Primary>
125 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
false
126
127 Token: operator Lexeme: +
128 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
129 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
130
131 Token: identifier Lexeme: floyd
132 <Term> -> <Factor> <Term Prime>
```

```
133 <Factor> -> - <Primary> | <Primary>
134 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
135
136 Token: operator Lexeme: >
137 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
138 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
139 <Relop> -> = | /= | > | < | => | <=
140
141 Token: identifier Lexeme: modest
142 <Expression> -> <Term> <Expression Prime>
143 <Term> -> <Factor> <Term Prime>
144 <Factor> -> - <Primary> | <Primary>
145 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
146
147 Token: operator Lexeme: /
148 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
149
150 Token: identifier Lexeme: mouse
151 <Factor> -> - <Primary> | <Primary>
152 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
153
154 Token: separator Lexeme: )
155 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
156 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
157
158 Token: identifier Lexeme: me
159 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
160 <Compound> -> { <Statement List> }
161 <Assign> -> <Identifier> := <Expression>;
162
163 Token: operator Lexeme: :=
164
165 Token: identifier Lexeme: alice
166 <Expression> -> <Term> <Expression Prime>
167 <Term> -> <Factor> <Term Prime>
168 <Factor> -> - <Primary> | <Primary>
169 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
170
171 Token: separator Lexeme: ;
172 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
173 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
174
175 Token: keyword Lexeme: else
176 <If> -> if (<Condition>) <Statement> endif |
177 if (<Condition>) <Statement> else <Statement> endif
178 <Return> -> return ; | return <Expression> ;
179 <Write> -> print (<Expression>);
180 <Read> -> read (<IDs>);
181 <While> -> while (<Condition>) <Statement>
182
183 Token: identifier Lexeme: me
184 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
185 <Compound> -> { <Statement List> }
186 <Assign> -> <Identifier> := <Expression>;
187
188 Token: operator Lexeme: :=
189
190 Token: identifier Lexeme: modest
191 <Expression> -> <Term> <Expression Prime>
192 <Term> -> <Factor> <Term Prime>
193 <Factor> -> - <Primary> | <Primary>
194 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
```

```
195
196 Token: operator Lexeme: +
197 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
198 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
199
200 Token: identifier Lexeme: mouse
201 <Term> -> <Factor> <Term Prime>
202 <Factor> -> - <Primary> | <Primary>
203 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
204
205 Token: separator Lexeme: ;
206 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
207 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
208
209 Token: keyword Lexeme: endif
210 <If> -> if (<Condition>) <Statement> endif |
211 if (<Condition>) <Statement> else <Statement> endif
212 <Return> -> return ; | return <Expression> ;
213 <Write> -> print (<Expression>);
214 <Read> -> read (<IDs>);
215 <While> -> while (<Condition>) <Statement>
216
217 Token: keyword Lexeme: return
218 <Return> -> return ; | return <Expression> ;
219
220 Token: identifier Lexeme: me
221 <Expression> -> <Term> <Expression Prime>
222 <Term> -> <Factor> <Term Prime>
223 <Factor> -> - <Primary> | <Primary>
224 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
    false
225
226 Token: separator Lexeme: [
227
228 Token: identifier Lexeme: modest
229 <IDs> -> <Identifier> | <Identifier>, <IDs>
230
231 Token: separator Lexeme: ,
232
233 Token: identifier Lexeme: alice
234
235 Token: separator Lexeme: ]
236
237 Token: separator Lexeme: ;
238 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
239 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
240
241 Token: separator Lexeme: }
242 <Write> -> print (<Expression>);
243 <Read> -> read (<IDs>);
244 <While> -> while (<Condition>) <Statement>
245 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
246 <Compound> -> { <Statement List> }
247 <Assign> -> <Identifier> := <Expression>;
248 <If> -> if (<Condition>) <Statement> endif |
249 if (<Condition>) <Statement> else <Statement> endif
250 <Return> -> return ; | return <Expression> ;
251 <Write> -> print (<Expression>);
252 <Read> -> read (<IDs>);
253 <While> -> while (<Condition>) <Statement>
254
255 Token: separator Lexeme: $$
256 <Function Definitions> -> <Function> | <Function> <Function Definitions>
257 <Function> -> function <Identifier> [ <Opt Parameter List> ] <Opt Declaration List> <Body>
258
259 Token: keyword Lexeme: while
```

```

260 <Opt Declaration List> -> <Declaration List> | <Empty>
261 <Declaration List> -> <Declaration> ; | <Declaration> ; <Declaration List>
262 <Declaration> -> <Qualifier> <IDs>
263 <Qualifier> -> integer | boolean | real
264 <Statement List> -> <Statement> | <Statement> <Statement List>
265 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
266 <Compound> -> { <Statement List> }
267 <Assign> -> <Identifier> := <Expression>;
268 <If> -> if (<Condition>) <Statement> endif |
269 if (<Condition>) <Statement> else <Statement> endif
270 <Return> -> return ; | return <Expression> ;
271 <Write> -> print (<Expression>);
272 <Read> -> read (<IDs>);
273 <While> -> while (<Condition>) <Statement>
274
275 Token: separator Lexeme: (
276
277 Token: integer Lexeme: 1
278 <Condition> -> <Expression> <Relop> <Expression>
279 <Expression> -> <Term> <Expression Prime>
280 <Term> -> <Factor> <Term Prime>
281 <Factor> -> - <Primary> | <Primary>
282 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
false
283
284 Token: operator Lexeme: >
285 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
286 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
287 <Relop> -> = | /= | > | < | => | <=
288
289 Token: integer Lexeme: 0
290 <Expression> -> <Term> <Expression Prime>
291 <Term> -> <Factor> <Term Prime>
292 <Factor> -> - <Primary> | <Primary>
293 <Primary> -> <Identifier> | <Integer> | <Identifier> [<IDs>] | (<Expression>) | <Real> | true |
false
294
295 Token: separator Lexeme: )
296 <Term Prime> -> * <Factor> <Term Prime> | / Factor <Term Prime> | epsilon
297 <Expression Prime> -> +<Term> <Expression Prime> | -<Term> <Expression Prime> | epsilon
298
299 Token: separator Lexeme: {
300 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
301 <Compound> -> { <Statement List> }
302
303 Token: keyword Lexeme: read
304 <Statement List> -> <Statement> | <Statement> <Statement List>
305 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
306 <Compound> -> { <Statement List> }
307 <Assign> -> <Identifier> := <Expression>;
308 <If> -> if (<Condition>) <Statement> endif |
309 if (<Condition>) <Statement> else <Statement> endif
310 <Return> -> return ; | return <Expression> ;
311 <Write> -> print (<Expression>);
312 <Read> -> read (<IDs>);
313
314 Token: separator Lexeme: (
315
316 Token: identifier Lexeme: Hello
317 <IDs> -> <Identifier> | <Identifier>, <IDs>
318
319 Token: separator Lexeme: )
320
321 Token: separator Lexeme: ;
322
323 Token: separator Lexeme: }
324 <While> -> while (<Condition>) <Statement>

```

```
325 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
326 <Compound> -> { <Statement List> }
327 <Assign> -> <Identifier> := <Expression>;
328 <If> -> if (<Condition>) <Statement> endif |
329 if (<Condition>) <Statement> else <Statement> endif
330 <Return> -> return ; | return <Expression> ;
331 <Write> -> print (<Expression>);
332 <Read> -> read (<IDs>);
333 <While> -> while (<Condition>) <Statement>
334
335 Token: separator Lexeme: $$
336 <Assign> -> <Identifier> := <Expression>;
337 <If> -> if (<Condition>) <Statement> endif |
338 if (<Condition>) <Statement> else <Statement> endif
339 <Return> -> return ; | return <Expression> ;
340 <Write> -> print (<Expression>);
341 <Read> -> read (<IDs>);
342 <While> -> while (<Condition>) <Statement>
343 <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Write> | <Read> | <While>
344 <Compound> -> { <Statement List> }
345 <Assign> -> <Identifier> := <Expression>;
346 <If> -> if (<Condition>) <Statement> endif |
347 if (<Condition>) <Statement> else <Statement> endif
348 <Return> -> return ; | return <Expression> ;
349 <Write> -> print (<Expression>);
350 <Read> -> read (<IDs>);
351 <While> -> while (<Condition>) <Statement>
352
353 Token: Lexeme:
```