

```
1:
2: Token: Keyword      Lexeme: function
3:
4:      <Rat18F> -> <Opt Function Definitions> $$ <Opt Declaration List> <Statemen
t List>
5:      <Opt Function Definitions> -> <Function Definitions> | <Empty>
6:
7: Token: Identifier    Lexeme: myInvalidFunction
8:
9:      <Function Definitions> -> <Function> | <Function> <Function Definitions>
10:     <Function> -> function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
11:     <Identifier>
12:
13: Token: Separator     Lexeme: (
14:
15:
16: Token: Identifier    Lexeme: var1x
17:
18:     <Opt Parameter List> -> <Parameter List> | <Empty>
19:     <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
20:     <Parameter> -> <IDs> : <Qualifier>
21:     <IDs> -> <Identifier> | <Identifier> , <IDs>
22:     <Identifier>
23:
24: Token: Separator     Lexeme: :
25:
26:
27: Token: Keyword      Lexeme: int
28:
29:     <Qualifier> -> int | boolean | real
30:
31: Token: Separator     Lexeme: )
32:
33:
34: Token: Separator     Lexeme: {
35:
36:     <Opt Declaration List> -> <Declaration List> | <Empty>
37:     <Empty> -> Îµ
38:     <Body> -> { <Statement List> }
39:
40: Token: Keyword      Lexeme: return
41:
42:     <Statement List> -> <Statement> | <Statement> <Statement List>
43:     <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
44:
45: Token: Separator     Lexeme: (
46:
47:     <Return> -> return; | return <Expression>;
48:     <Expression> -> <Term> <ExpressionPrime>
49:     <Term> -> <Factor> <TermPrime>
50:     <Factor> -> - <Primary> | <Primary>
51:     <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
52:
53: Token: Identifier    Lexeme: var1x
54:
55:     <Expression> -> <Term> <ExpressionPrime>
56:     <Term> -> <Factor> <TermPrime>
57:     <Factor> -> - <Primary> | <Primary>
58:     <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
59:     <Identifier>
60:
61: Token: Operator      Lexeme: *
62:
63:     <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
64:
65: Token: Identifier    Lexeme: var2x
```

```

66:
67:         <Factor> -> - <Primary> | <Primary>
68:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
e> sion> ) | <Real> | true | false
69:         <Identifier>
70:
71: Token:   Separator           Lexeme:  )
72:
73:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
74:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
75:         <Empty> -> Îµ
76:
77: Token:   Operator           Lexeme:  +
78:
79:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
80:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
81:
82: Token:   Integer           Lexeme:  50
83:
84:         <Term> -> <Factor> <TermPrime>
85:         <Factor> -> - <Primary> | <Primary>
86:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
e> sion> ) | <Real> | true | false
87:         <Integer>
88:
89: Token:   Separator           Lexeme:  ;
90:
91:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
92:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
93:         <Empty> -> Îµ
94:
95: Token:   Separator           Lexeme:  }
96:
97:
98: Token:   Separator           Lexeme:  $$
99:
100:
101: Token:   Keyword           Lexeme:  int
102:
103:         <Opt Declaration List> -> <Declaration List> | <Empty>
104:         <Declaration List> -> <Declaration>; | <Declaration>; <Declaration List>
105:         <Declaration> -> <Qualifier> <IDs>
106:         <Qualifier> -> int | boolean | real
107:
108: Token:   Identifier         Lexeme:  a
109:
110:         <IDs> -> <Identifier> | <Identifier>, <IDs>
111:         <Identifier>
112:
113: Token:   Separator           Lexeme:  ,
114:
115:
116: Token:   Identifier         Lexeme:  b
117:
118:         <IDs> -> <Identifier> | <Identifier>, <IDs>
119:         <Identifier>
120:
121: Token:   Separator           Lexeme:  ;
122:
123:
124: Token:   Identifier         Lexeme:  a
125:
126:         <Statement List> -> <Statement> | <Statement> <Statement List>
127:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
128:         <Assign> -> <Identifier> = <Expression>;
129:         <Identifier>

```

```
130:
131: Token:  Operator          Lexeme: =
132:
133:
134: Token:  Integer          Lexeme: 0
135:
136:         <Expression> -> <Term> <ExpressionPrime>
137:         <Term> -> <Factor> <TermPrime>
138:         <Factor> -> - <Primary> | <Primary>
139:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
140:         <Integer>
141:
142: Token:  Separator        Lexeme: ;
143:
144:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
145:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
146:         <Empty> -> Îµ
147:
148: Token:  Identifier       Lexeme: b
149:
150:         <Statement List> -> <Statement> | <Statement> <Statement List>
151:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
152:         <Assign> -> <Identifier> = <Expression>;
153:         <Identifier>
154:
155: Token:  Operator          Lexeme: =
156:
157:
158: Token:  Integer          Lexeme: 1
159:
160:         <Expression> -> <Term> <ExpressionPrime>
161:         <Term> -> <Factor> <TermPrime>
162:         <Factor> -> - <Primary> | <Primary>
163:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
164:         <Integer>
165:
166: Token:  Separator        Lexeme: ;
167:
168:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
169:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
170:         <Empty> -> Îµ
171:
172: Token:  Illegal          Lexeme: invalidVar1
173:
174:
175: ERROR: Illegal symbol 'invalidVar1' Line: 15
```