```
1
2  //----------------------------------------------------
3  // The following code was generated by CUP v0.10k
4  // Sat Apr 30 10:55:19 PDT 2005
5  //----------------------------------------------------
6
7  import java_cup.runtime.*;
8
9  /** CUP v0.10k generated parser.
10   * @version Sat Apr 30 10:55:19 PDT 2005
11   */
12 public class EJayParser extends java_cup.runtime.lr_parser {
13
14   /** Default constructor. */
15   public EJayParser() {super();}
16
17   /** Constructor which sets the default scanner. */
18   public EJayParser(java_cup.runtime.Scanner s) {super(s);}
19
20   /** Production table. */
21   protected static final short _production_table[][] =
22     unpackFromStrings(new String[] {
23     "\000\144\000\002\003\003\000\002\002\004\000\002\004" +
24     "\003\000\002\004\004\000\002\005\004\000\002\005\003" +
25     "\000\002\006\004\000\002\013\003\000\002\013\003\000" +
26     "\002\013\003\000\002\013\003\000\002\013\003\000\002" +
27     "\013\003\000\002\013\003\000\002\015\006\000\002\020" +
28     "\002\000\002\020\003\000\002\020\005\000\002\016\006" +
29     "\000\002\017\003\000\002\023\004\000\002\023\005\000" +
30     "\002\024\003\000\002\010\007\000\002\011\004\000\002" +
31     "\012\004\000\002\012\005\000\002\012\006\000\002\025" +
32     "\003\000\002\025\005\000\002\030\002\000\002\030\004" +
33     "\000\002\030\005\000\002\026\004\000\002\026\005\000" +
34     "\002\026\006\000\002\027\003\000\002\007\004\000\002" +
35     "\007\005\000\002\031\003\000\002\031\004\000\002\032" +
36     "\003\000\002\032\003\000\002\032\003\000\002\032\003" +
37     "\000\002\032\003\000\002\032\003\000\002\032\003\000" +
38     "\002\032\003\000\002\033\006\000\002\041\003\000\002" +
39     "\041\003\000\002\041\003\000\002\043\006\000\002\044" +
40     "\005\000\002\034\007\000\002\034\011\000\002\035\007" +
41     "\000\002\036\006\000\002\036\007\000\002\037\005\000" +
42     "\002\040\005\000\002\042\005\000\002\042\005\000\002" +
43     "\042\005\000\002\042\005\000\002\042\005\000\002\042" +
44     "\004\000\002\042\005\000\002\042\006\000\002\042\003" +
45     "\000\002\042\003\000\002\042\005\000\002\046\003\000" +
46     "\002\046\003\000\002\046\003\000\002\046\003\000\002" +
47     "\046\003\000\002\046\003\000\002\047\003\000\002\047" +
48     "\003\000\002\050\003\000\002\050\003\000\002\051\003" +
49     "\000\002\051\003\000\002\051\003\000\002\045\003\000" +
50     "\002\045\005\000\002\014\003\000\002\014\005\000\002" +
51     "\052\003\000\002\052\003\000\002\052\003\000\002\052" +
52     "\003\000\002\021\003\000\002\022\003\000\002\053\003" +
53     "\000\002\054\003\000\002\055\003\000\002\055\003" });
54
55   /** Access to production table. */
56   public short[][] production_table() {return _production_table;}
57
58   /** Parse-action table. */
59   protected static final short[][] _action_table =
60     unpackFromStrings(new String[] {
61     "\000\243\000\016\004\011\007\005\010\012\014\023\015" +
62     "\015\016\010\001\002\000\006\030\036\053\040\001\002" +
63     "\000\006\030\ufffa\053\ufffa\001\002\000\004\022\053\001" +
64     "\002\000\004\002\052\001\002\000\004\024\uffee\001\002" +
65     "\000\006\030\ufff7\053\ufff7\001\002\000\006\030\ufff6\053" +
66     "\ufff6\001\002\000\004\024\026\001\002\000\020\002\uffff" +
67     "\004\011\007\005\010\012\014\023\015\015\016\010\001" +
68     "\002\000\006\030\ufff8\053\ufff8\001\002\000\004\002\001" +
69     "\001\002\000\006\030\ufff4\053\ufff4\001\002\000\020\002" +
70     "\ufffc\004\ufffc\007\ufffc\010\ufffc\014\ufffc\015\ufffc\016\ufffc" +
71     "\001\002\000\006\030\ufff5\053\ufff5\001\002\000\004\026" +
72     "\024\001\002\000\006\030\ufff9\053\ufff9\001\002\000\020" +
73     "\002\ufffd\004\ufffd\007\ufffd\010\ufffd\014\ufffd\015\ufffd\016" +
74     "\ufffd\001\002\000\004\002\ufffe\001\002\000\016\004\011" +
75     "\007\005\010\012\014\023\015\015\016\010\001\002\000" +
76     "\006\030\036\053\040\001\002\000\004\025\035\001\002" +
77     "\000\004\026\033\001\002\000\004\026\uffeb\001\002\000" +
78     "\020\004\011\007\005\010\012\014\023\015\015\016\010" +
79     "\025\uffed\001\002\000\004\025\uffec\001\002\000\006\030" +
80     "\uffef\053\uffef\001\002\000\006\031\ufff2\054\044\001\002" +
81     "\000\004\026\ufffb\001\002\000\052\022\uffa3\023\uffa3\026" +
82     "\uffa3\027\uffa3\030\uffa3\031\uffa3\032\uffa3\033\uffa3\034\uffa3" +
83     "\035\uffa3\036\uffa3\037\uffa3\040\uffa3\041\uffa3\042\uffa3\043" +
84     "\uffa3\044\uffa3\045\uffa3\046\uffa3\050\uffa3\001\002\000\006" +
85     "\026\uffa9\027\042\001\002\000\004\053\040\001\002\000" +
86     "\004\026\uffa8\001\002\000\042\023\uffa2\026\uffa2\027\uffa2" +
87     "\031\uffa2\033\uffa2\034\uffa2\035\uffa2\036\uffa2\037\uffa2\040" +
88     "\uffa2\041\uffa2\042\uffa2\043\uffa2\044\uffa2\045\uffa2\046\uffa2" +
89     "\001\002\000\006\027\050\031\ufff1\001\002\000\004\031" +
90     "\047\001\002\000\006\030\ufff3\053\ufff3\001\002\000\006" +
91     "\031\ufff2\054\044\001\002\000\004\031\ufff0\001\002\000" +
92     "\004\002\000\001\002\000\024\004\011\007\005\010\012" +
93     "\014\023\015\015\016\010\021\057\023\uffe3\027\uffe3\001" +
94     "\002\000\006\030\036\053\040\001\002\000\004\023\064" +
95     "\001\002\000\006\023\uffe5\027\062\001\002\000\016\004" +
96     "\011\007\005\010\012\014\023\015\015\016\010\001\002" +
97     "\000\006\030\036\053\040\001\002\000\006\023\uffe1\027" +
98     "\uffe1\001\002\000\024\004\011\007\005\010\012\014\023" +
99     "\015\015\016\010\021\057\023\uffe3\027\uffe3\001\002\000" +
100    "\004\023\uffe4\001\002\000\004\024\066\001\002\000\020" +
101    "\002\uffea\004\uffea\007\uffea\010\uffea\014\uffea\015\uffea\016" +
102    "\uffea\001\002\000\036\004\011\006\104\007\005\010\012" +
103    "\011\073\014\023\015\015\016\010\017\101\020\110\024" +
104    "\072\025\113\026\074\053\040\001\002\000\010\030\204" +
105    "\032\241\050\205\001\002\000\024\005\uffd6\006\uffd6\011" +
106    "\uffd6\017\uffd6\020\uffd6\024\uffd6\025\uffd6\026\uffd6\053\uffd6" +
107    "\001\002\000\004\026\237\001\002\000\036\004\uffdd\006" +
108    "\104\007\uffdd\010\uffdd\011\073\014\uffdd\015\uffdd\016\uffdd" +
109    "\017\101\020\110\024\072\025\235\026\074\053\040\001" +
110    "\002\000\004\022\230\001\002\000\024\005\uffd8\006\uffd8" +
111    "\011\uffd8\017\uffd8\020\uffd8\024\uffd8\025\uffd8\026\uffd8\053" +
112    "\uffd8\001\002\000\024\005\uffd2\006\uffd2\011\uffd2\017\uffd2" +
```

```
    "\020\uffd2\024\uffd2\025\uffd2\026\uffd2\053\uffd2\001\002\000" +
    "\024\005\uffd1\006\uffd1\011\uffd1\017\uffd1\020\uffd1\024\uffd1" +
    "\025\uffd1\026\uffd1\053\uffd1\001\002\000\004\025\227\001" +
    "\002\000\024\005\uffd3\006\uffd3\011\uffd3\017\uffd3\020\uffd3" +
    "\024\uffd3\025\uffd3\026\uffd3\053\uffd3\001\002\000\026\012" +
    "\132\013\135\022\127\047\131\051\144\052\140\053\040" +
    "\054\044\055\141\056\125\001\002\000\050\023\uffce\026" +
    "\uffce\027\uffce\030\uffce\031\uffce\032\uffce\033\uffce\034\uffce" +
    "\035\uffce\036\uffce\037\uffce\040\uffce\041\uffce\042\uffce\043" +
    "\uffce\044\uffce\045\uffce\046\uffce\050\uffce\001\002\000\024" +
    "\005\uffd5\006\uffd5\011\uffd5\017\uffd5\020\uffd5\024\uffd5\025" +
    "\uffd5\026\uffd5\053\uffd5\001\002\000\004\022\217\001\002" +
    "\000\024\005\uffd4\006\uffd4\011\uffd4\017\uffd4\020\uffd4\024" +
    "\uffd4\025\uffd4\026\uffd4\053\uffd4\001\002\000\012\022\211" +
    "\030\uffcf\032\uffcf\050\uffcf\001\002\000\024\005\uffd7\006" +
    "\uffd7\011\uffd7\017\uffd7\020\uffd7\024\uffd7\025\uffd7\026\uffd7" +
    "\053\uffd7\001\002\000\026\012\132\013\135\022\127\047" +
    "\131\051\144\052\140\053\040\054\044\055\141\056\125" +
    "\001\002\000\022\006\104\011\073\017\101\020\110\024" +
    "\072\025\uffda\026\074\053\040\001\002\000\016\004\011" +
    "\007\005\010\012\014\023\015\015\016\010\001\002\000" +
    "\020\002\uffe8\004\uffe8\007\uffe8\010\uffe8\014\uffe8\015\uffe8" +
    "\016\uffe8\001\002\000\020\006\104\011\073\017\101\020" +
    "\110\024\072\026\074\053\040\001\002\000\050\023\uffcd" +
    "\026\uffcd\027\uffcd\030\uffcd\031\uffcd\032\uffcd\033\uffcd\034" +
    "\uffcd\035\uffcd\036\uffcd\037\uffcd\040\uffcd\041\uffcd\042\uffcd" +
    "\043\uffcd\044\uffcd\045\uffcd\046\uffcd\050\uffcd\001\002\000" +
    "\004\025\117\001\002\000\020\002\uffe6\004\uffe6\007\uffe6" +
    "\010\uffe6\014\uffe6\015\uffe6\016\uffe6\001\002\000\020\006" +
    "\104\011\073\017\101\020\110\024\072\026\074\053\040" +
    "\001\002\000\004\025\122\001\002\000\024\005\uffde\006" +
    "\uffde\011\uffde\017\uffde\020\uffde\024\uffde\025\uffde\026\uffde" +
    "\053\uffde\001\002\000\004\025\uffd9\001\002\000\046\023" +
    "\uffbb\026\uffbb\027\uffbb\030\204\031\uffbb\033\uffbb\034\uffbb" +
    "\035\uffbb\036\uffbb\037\uffbb\040\uffbb\041\uffbb\042\uffbb\043" +
    "\uffbb\044\uffbb\045\uffbb\046\uffbb\050\205\001\002\000\042" +
    "\023\uffa0\026\uffa0\027\uffa0\031\uffa0\033\uffa0\034\uffa0\035" +
    "\uffa0\036\uffa0\037\uffa0\040\uffa0\041\uffa0\042\uffa0\043\uffa0" +
    "\044\uffa0\045\uffa0\046\uffa0\001\002\000\004\026\203\001" +
    "\002\000\026\012\132\013\135\022\127\047\131\051\144" +
    "\052\140\053\040\054\044\055\141\056\125\001\002\000" +
    "\042\023\uffa6\026\uffa6\027\uffa6\031\uffa6\033\uffa6\034\uffa6" +
    "\035\uffa6\036\uffa6\037\uffa6\040\uffa6\041\uffa6\042\uffa6\043" +
    "\uffa6\044\uffa6\045\uffa6\046\uffa6\001\002\000\026\012\uffac" +
    "\013\uffac\022\uffac\047\uffac\051\uffac\052\uffac\053\uffac\054" +
    "\uffac\055\uffac\056\uffac\001\002\000\042\023\uff9f\026\uff9f" +
    "\027\uff9f\031\uff9f\033\uff9f\034\uff9f\035\uff9f\036\uff9f\037" +
    "\uff9f\040\uff9f\041\uff9f\042\uff9f\043\uff9f\044\uff9f\045\uff9f" +
    "\046\uff9f\001\002\000\042\023\uffa7\026\uffa7\027\uffa7\031" +
    "\uffa7\033\uffa7\034\uffa7\035\uffa7\036\uffa7\037\uffa7\040\uffa7" +
    "\041\uffa7\042\uffa7\043\uffa7\044\uffa7\045\uffa7\046\uffa7\001" +
    "\002\000\050\022\175\023\uffcf\026\uffcf\027\uffcf\030\uffcf" +
    "\031\uffcf\033\uffcf\034\uffcf\035\uffcf\036\uffcf\037\uffcf\040" +
    "\uffcf\041\uffcf\042\uffcf\043\uffcf\044\uffcf\045\uffcf\046\uffcf" +
    "\050\uffcf\001\002\000\042\023\uff9e\026\uff9e\027\uff9e\031" +
    "\uff9e\033\uff9e\034\uff9e\035\uff9e\036\uff9e\037\uff9e\040\uff9e" +
    "\041\uff9e\042\uff9e\043\uff9e\044\uff9e\045\uff9e\046\uff9e\001" +
    "\002\000\026\012\132\013\135\022\127\047\131\051\144" +
    "\052\140\053\040\054\044\055\141\056\125\001\002\000" +
    "\042\023\uffba\026\uffba\027\uffba\031\uffba\033\uffba\034\uffba" +
    "\035\uffba\036\uffba\037\uffba\040\uffba\041\uffba\042\uffba\043" +
    "\uffba\044\uffba\045\uffba\046\uffba\001\002\000\026\012\uffad" +
    "\013\uffad\022\uffad\047\uffad\051\uffad\052\uffad\053\uffad\054" +
    "\uffad\055\uffad\056\uffad\001\002\000\042\023\uffa1\026\uffa1" +
    "\027\uffa1\031\uffa1\033\uffa1\034\uffa1\035\uffa1\036\uffa1\037" +
    "\uffa1\040\uffa1\041\uffa1\042\uffa1\043\uffa1\044\uffa1\045\uffa1" +
    "\046\uffa1\001\002\000\042\023\uffa5\026\uffa5\027\uffa5\031" +
    "\uffa5\033\uffa5\034\uffa5\035\uffa5\036\uffa5\037\uffa5\040\uffa5" +
    "\041\uffa5\042\uffa5\043\uffa5\044\uffa5\045\uffa5\046\uffa5\001" +
    "\002\000\042\023\uffab\026\uffab\027\157\031\uffab\033\154" +
    "\034\147\035\162\036\151\037\160\040\156\041\155\042" +
    "\146\043\165\044\152\045\150\046\164\001\002\000\026" +
    "\012\uffae\013\uffae\022\uffae\047\uffae\051\uffae\052\uffae\053" +
    "\uffae\054\uffae\055\uffae\056\uffae\001\002\000\042\023\uffa4" +
    "\026\uffa4\027\uffa4\031\uffa4\033\uffa4\034\uffa4\035\uffa4\036" +
    "\uffa4\037\uffa4\040\uffa4\041\uffa4\042\uffa4\043\uffa4\044\uffa4" +
    "\045\uffa4\046\uffa4\001\002\000\026\012\uffb5\013\uffb5\022" +
    "\uffb5\047\uffb5\051\uffb5\052\uffb5\053\uffb5\054\uffb5\055\uffb5" +
    "\056\uffb5\001\002\000\026\012\uffb1\013\uffb1\022\uffb1\047" +
    "\uffb1\051\uffb1\052\uffb1\053\uffb1\054\uffb1\055\uffb1\056\uffb1" +
    "\001\002\000\026\012\132\013\135\022\127\047\131\051" +
    "\144\052\140\053\040\054\044\055\141\056\125\001\002" +
    "\000\026\012\uffaf\013\uffaf\022\uffaf\047\uffaf\051\uffaf\052" +
    "\uffaf\053\uffaf\054\uffaf\055\uffaf\056\uffaf\001\002\000\026" +
    "\012\uffb3\013\uffb3\022\uffb3\047\uffb3\051\uffb3\052\uffb3\053" +
    "\uffb3\054\uffb3\055\uffb3\056\uffb3\001\002\000\026\012\132" +
    "\013\135\022\127\047\131\051\144\052\140\053\040\054" +
    "\044\055\141\056\125\001\002\000\026\012\uffb2\013\uffb2" +
    "\022\uffb2\047\uffb2\051\uffb2\052\uffb2\053\uffb2\054\uffb2\055" +
    "\uffb2\056\uffb2\001\002\000\026\012\uffb6\013\uffb6\022\uffb6" +
    "\047\uffb6\051\uffb6\052\uffb6\053\uffb6\054\uffb6\055\uffb6\056" +
    "\uffb6\001\002\000\026\012\uffb7\013\uffb7\022\uffb7\047\uffb7" +
    "\051\uffb7\052\uffb7\053\uffb7\054\uffb7\055\uffb7\056\uffb7\001" +
    "\002\000\026\012\132\013\135\022\127\047\131\051\144" +
    "\052\140\053\040\054\044\055\141\056\125\001\002\000" +
    "\026\012\uffb8\013\uffb8\022\uffb8\047\uffb8\051\uffb8\052\uffb8" +
    "\053\uffb8\054\uffb8\055\uffb8\056\uffb8\001\002\000\026\012" +
    "\132\013\135\022\127\047\131\051\144\052\140\053\040" +
    "\054\044\055\141\056\125\001\002\000\026\012\uffb0\013" +
    "\uffb0\022\uffb0\047\uffb0\051\uffb0\052\uffb0\053\uffb0\054\uffb0" +
    "\055\uffb0\056\uffb0\001\002\000\026\012\132\013\135\022" +
    "\127\047\131\051\144\052\140\053\040\054\044\055\141" +
    "\056\125\001\002\000\026\012\132\013\135\022\127\047" +
    "\131\051\144\052\140\053\040\054\044\055\141\056\125" +
    "\001\002\000\026\012\uffb4\013\uffb4\022\uffb4\047\uffb4\051" +
    "\uffb4\052\uffb4\053\uffb4\054\uffb4\055\uffb4\056\uffb4\001\002" +
    "\000\042\023\uffc3\026\uffc3\027\uffc3\031\uffc3\033\154\034" +
    "\147\035\162\036\151\037\160\040\156\041\155\042\146" +
    "\043\165\044\152\045\150\046\uffc3\001\002\000\042\023" +
    "\uffc1\026\uffc1\027\uffc1\031\uffc1\033\154\034\147\035\162" +
    "\036\151\037\uffc1\040\uffc1\041\uffc1\042\uffc1\043\uffc1\044" +
    "\uffc1\045\uffc1\046\uffc1\001\002\000\042\023\uffbf\026\uffbf" +
```

```
    "\027\uffbf\031\uffbf\033\uffbf\034\uffbf\035\uffbf\036\uffbf\037" +
    "\uffbf\040\uffbf\041\uffbf\042\uffbf\043\uffbf\044\uffbf\045\uffbf" +
    "\046\uffbf\001\002\000\010\023\uffaa\026\uffaa\031\uffaa\001" +
    "\002\000\042\023\uffc0\026\uffc0\027\uffc0\031\uffc0\033\uffc0" +
    "\034\uffc0\035\162\036\151\037\uffc0\040\uffc0\041\uffc0\042" +
    "\uffc0\043\uffc0\044\uffc0\045\uffc0\046\uffc0\001\002\000\042" +
    "\023\uffc2\026\uffc2\027\uffc2\031\uffc2\033\154\034\147\035" +
    "\162\036\151\037\160\040\156\041\155\042\146\043\165" +
    "\044\152\045\uffc2\046\uffc2\001\002\000\042\023\uffbe\026" +
    "\uffbe\027\uffbe\031\uffbe\033\uffbe\034\uffbe\035\uffbe\036\uffbe" +
    "\037\uffbe\040\uffbe\041\uffbe\042\uffbe\043\uffbe\044\uffbe\045" +
    "\uffbe\046\uffbe\001\002\000\030\012\132\013\135\022\127" +
    "\023\177\047\131\051\144\052\140\053\040\054\044\055" +
    "\141\056\125\001\002\000\004\023\200\001\002\000\042" +
    "\023\uffbd\026\uffbd\027\uffbd\031\uffbd\033\uffbd\034\uffbd\035" +
    "\uffbd\036\uffbd\037\uffbd\040\uffbd\041\uffbd\042\uffbd\043\uffbd" +
    "\044\uffbd\045\uffbd\046\uffbd\001\002\000\042\023\uffbc\026" +
    "\uffbc\027\uffbc\031\uffbc\033\uffbc\034\uffbc\035\uffbc\036\uffbc" +
    "\037\uffbc\040\uffbc\041\uffbc\042\uffbc\043\uffbc\044\uffbc\045" +
    "\uffbc\046\uffbc\001\002\000\034\023\202\033\154\034\147" +
    "\035\162\036\151\037\160\040\156\041\155\042\146\043" +
    "\165\044\152\045\150\046\164\001\002\000\042\023\uffb9" +
    "\026\uffb9\027\uffb9\031\uffb9\033\uffb9\034\uffb9\035\uffb9\036" +
    "\uffb9\037\uffb9\040\uffb9\041\uffb9\042\uffb9\043\uffb9\044\uffb9" +
    "\045\uffb9\046\uffb9\001\002\000\024\005\uffc4\006\uffc4\011" +
    "\uffc4\017\uffc4\020\uffc4\024\uffc4\025\uffc4\026\uffc4\053\uffc4" +
    "\001\002\000\026\012\132\013\135\022\127\047\131\051" +
    "\144\052\140\053\040\054\044\055\141\056\125\001\002" +
    "\000\004\053\040\001\002\000\050\023\uffcb\026\uffcb\027" +
    "\uffcb\030\uffcb\031\uffcb\032\uffcb\033\uffcb\034\uffcb\035\uffcb" +
    "\036\uffcb\037\uffcb\040\uffcb\041\uffcb\042\uffcb\043\uffcb\044" +
    "\uffcb\045\uffcb\046\uffcb\050\uffcb\001\002\000\004\031\210" +
    "\001\002\000\050\023\uffcc\026\uffcc\027\uffcc\030\uffcc\031" +
    "\uffcc\032\uffcc\033\uffcc\034\uffcc\035\uffcc\036\uffcc\037\uffcc" +
    "\040\uffcc\041\uffcc\042\uffcc\043\uffcc\044\uffcc\045\uffcc\046" +
    "\uffcc\050\uffcc\001\002\000\030\012\132\013\135\022\127" +
    "\023\213\047\131\051\144\052\140\053\040\054\044\055" +
    "\141\056\125\001\002\000\004\023\215\001\002\000\004" +
    "\026\214\001\002\000\024\005\uffc7\006\uffc7\011\uffc7\017" +
    "\uffc7\020\uffc7\024\uffc7\025\uffc7\026\uffc7\053\uffc7\001\002" +
    "\000\004\026\216\001\002\000\024\005\uffc6\006\uffc6\011" +
    "\uffc6\017\uffc6\020\uffc6\024\uffc6\025\uffc6\026\uffc6\053\uffc6" +
    "\001\002\000\026\012\132\013\135\022\127\047\131\051" +
    "\144\052\140\053\040\054\044\055\141\056\125\001\002" +
    "\000\034\023\221\033\154\034\147\035\162\036\151\037" +
    "\160\040\156\041\155\042\146\043\165\044\152\045\150" +
    "\046\164\001\002\000\020\006\104\011\073\017\101\020" +
    "\110\024\072\026\074\053\040\001\002\000\024\005\223" +
    "\006\uffca\011\uffca\017\uffca\020\uffca\024\uffca\025\uffca\026" +
    "\uffca\053\uffca\001\002\000\020\006\104\011\073\017\101" +
    "\020\110\024\072\026\074\053\040\001\002\000\024\005" +
    "\uffc9\006\uffc9\011\uffc9\017\uffc9\020\uffc9\024\uffc9\025\uffc9" +
    "\026\uffc9\053\uffc9\001\002\000\034\026\226\033\154\034" +
    "\147\035\162\036\151\037\160\040\156\041\155\042\146" +
    "\043\165\044\152\045\150\046\164\001\002\000\024\005" +
    "\uffc5\006\uffc5\011\uffc5\017\uffc5\020\uffc5\024\uffc5\025\uffc5" +
    "\026\uffc5\053\uffc5\001\002\000\020\002\uffe7\004\uffe7\007" +
    "\uffe7\010\uffe7\014\uffe7\015\uffe7\016\uffe7\001\002\000\026" +
    "\012\132\013\135\022\127\047\131\051\144\052\140\053" +
    "\040\054\044\055\141\056\125\001\002\000\034\023\232" +
    "\033\154\034\147\035\162\036\151\037\160\040\156\041" +
    "\155\042\146\043\165\044\152\045\150\046\164\001\002" +
    "\000\020\006\104\011\073\017\101\020\110\024\072\026" +
    "\074\053\040\001\002\000\024\005\uffc8\006\uffc8\011\uffc8" +
    "\017\uffc8\020\uffc8\024\uffc8\025\uffc8\026\uffc8\053\uffc8\001" +
    "\002\000\004\025\236\001\002\000\024\005\uffe0\006\uffe0" +
    "\011\uffe0\017\uffe0\020\uffe0\024\uffe0\025\uffe0\026\uffe0\053" +
    "\uffe0\001\002\000\024\005\uffdf\006\uffdf\011\uffdf\017\uffdf" +
    "\020\uffdf\024\uffdf\025\uffdf\026\uffdf\053\uffdf\001\002\000" +
    "\034\004\011\006\uffdc\007\005\010\012\011\uffdc\014\023" +
    "\015\015\016\010\017\uffdc\020\uffdc\024\uffdc\026\uffdc\053" +
    "\uffdc\001\002\000\020\006\uffdb\011\uffdb\017\uffdb\020\uffdb" +
    "\024\uffdb\026\uffdb\053\uffdb\001\002\000\026\012\132\013" +
    "\135\022\127\047\131\051\144\052\140\053\040\054\044" +
    "\055\141\056\125\001\002\000\034\026\243\033\154\034" +
    "\147\035\162\036\151\037\160\040\156\041\155\042\146" +
    "\043\165\044\152\045\150\046\164\001\002\000\024\005" +
    "\uffd0\006\uffd0\011\uffd0\017\uffd0\020\uffd0\024\uffd0\025\uffd0" +
    "\026\uffd0\053\uffd0\001\002\000\006\023\uffe2\027\uffe2\001" +
    "\002\000\010\022\uffe9\026\uffa9\027\042\001\002" });

  /** Access to parse-action table. */
  public short[][] action_table() {return _action_table;}

  /** <code>reduce_goto</code> table. */
  protected static final short[][] _reduce_table =
    unpackFromStrings(new String[] {
    "\000\243\000\026\003\006\004\015\005\013\006\021\010" +
    "\017\011\005\013\003\015\020\016\016\017\012\001\001" +
    "\000\006\014\036\021\244\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\002\001\001\000\024\004" +
    "\024\005\013\006\021\010\017\011\005\013\003\015\020" +
    "\016\016\017\012\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
    "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
    "\002\001\001\000\020\006\031\013\026\015\020\016\016" +
    "\017\012\023\027\024\030\001\001\000\006\014\036\021" +
    "\040\001\001\000\002\001\001\000\002\001\001\000\002" +
    "\001\001\000\020\006\031\013\026\015\020\016\016\017" +
    "\012\023\033\024\030\001\001\000\002\001\001\000\002" +
    "\001\001\000\006\020\045\022\044\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\006\014\042" +
    "\021\040\001\001\000\002\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\006" +
    "\020\050\022\044\001\001\000\002\001\001\000\002\001" +
    "\001\000\016\013\053\015\020\016\016\017\012\025\054" +
    "\030\055\001\001\000\004\021\243\001\001\000\002\001" +
    "\001\000\002\001\001\000\012\013\057\015\020\016\016" +
    "\017\012\001\001\000\004\021\060\001\001\000\002\001" +
    "\001\000\016\013\053\015\020\016\016\017\012\025\062" +
    "\030\055\001\001\000\002\001\001\000\004\012\064\001" +
```

```
    "\001\000\002\001\001\000\052\006\070\007\113\013\026" +
    "\015\020\016\016\017\012\021\105\026\106\027\111\031" +
    "\076\032\110\033\067\034\102\035\104\036\077\037\074" +
    "\040\075\041\066\043\101\044\114\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\036\021\105" +
    "\026\106\027\111\031\233\032\110\033\067\034\102\035" +
    "\104\036\077\037\074\040\075\041\066\043\101\044\114" +
    "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
    "\000\030\021\133\022\132\041\123\042\224\043\101\044" +
    "\114\051\135\052\136\053\127\054\141\055\144\001\001" +
    "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\032" +
    "\021\133\022\132\041\123\042\142\043\101\044\114\045" +
    "\175\051\135\052\136\053\127\054\141\055\144\001\001" +
    "\125\051\135\052\136\053\127\054\141\055\144\001\001" +
    "\000\036\021\105\026\106\027\111\031\122\032\110\033" +
    "\067\034\102\035\104\036\077\037\074\040\075\041\066" +
    "\043\101\044\114\001\001\000\016\006\070\007\117\013" +
    "\026\015\020\016\016\017\012\001\001\000\002\001\001" +
    "\000\036\021\105\026\106\027\111\031\115\032\110\033" +
    "\067\034\102\035\104\036\077\037\074\040\075\041\066" +
    "\043\101\044\114\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\036\021\105\026\106\027\111" +
    "\031\120\032\110\033\067\034\102\035\104\036\077\037" +
    "\074\040\075\041\066\043\101\044\114\001\001\000\002" +
    "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\030\021\133" +
    "\022\132\041\123\042\200\043\101\044\114\051\135\052" +
    "\136\053\127\054\141\055\144\001\001\000\002\001\001" +
    "\000\002\001\001\000\002\001\001\000\030\021\133\022" +
    "\041\123\042\173\043\101\044\114\051\135\052\136\053" +
    "\127\054\141\055\144\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\002\001\001\000\010\046" +
    "\162\047\152\050\160\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\002\001\001\000\030\021" +
    "\133\022\132\041\123\042\172\043\101\044\114\051\135" +
    "\052\136\053\127\054\141\055\144\001\001\000\002\001" +
    "\001\000\002\001\001\000\030\021\133\022\132\041\123" +
    "\042\171\043\101\044\114\051\135\052\136\053\127\054" +
    "\141\055\144\001\001\000\002\001\001\000\002\001\001" +
    "\000\002\001\001\000\032\021\133\022\132\041\123\042" +
    "\142\043\101\044\114\045\170\051\135\052\136\053\127" +
    "\054\141\055\144\001\001\000\002\001\001\000\030\021" +
    "\133\022\132\041\123\042\167\043\101\044\114\051\135" +
    "\052\136\053\127\054\141\055\144\001\001\000\002\001" +
    "\001\000\030\021\133\022\132\041\123\042\166\043\101" +
    "\044\114\051\135\052\136\053\127\054\141\055\144\001" +
    "\001\000\030\021\133\022\132\041\123\042\165\043\101" +
    "\044\114\051\135\052\136\053\127\054\141\055\144\001" +
    "\001\000\002\001\001\000\010\046\162\047\152\050\160" +
    "\001\001\000\010\046\162\047\152\050\160\001\001\000" +
    "\010\046\162\047\152\050\160\001\001\000\002\001\001" +
    "\000\010\046\162\047\152\050\160\001\001\000\010\046" +
    "\162\047\152\050\160\001\001\000\010\046\162\047\152" +
    "\050\160\001\001\000\032\021\133\022\132\041\123\042" +
    "\142\043\101\044\114\045\175\051\135\052\136\053\127" +
    "\054\141\055\144\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\010\046\162\047\152\050\160" +
    "\001\001\000\002\001\001\000\002\001\001\000\032\021" +
    "\133\022\132\041\123\042\142\043\101\044\114\045\206" +
    "\051\135\052\136\053\127\054\141\055\144\001\001\000" +
    "\004\021\205\001\001\000\002\001\001\000\002\001\001" +
    "\000\002\001\001\000\032\021\133\022\132\041\123\042" +
    "\142\043\101\044\114\045\211\051\135\052\136\053\127" +
    "\054\141\055\144\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
    "\000\030\021\133\022\132\041\123\042\217\043\101\044" +
    "\114\051\135\052\136\053\127\054\141\055\144\001\001" +
    "\000\010\046\162\047\152\050\160\001\001\000\034\021" +
    "\105\026\106\027\111\032\221\033\067\034\102\035\104" +
    "\036\077\037\074\040\075\041\066\043\101\044\114\001" +
    "\001\000\002\001\001\000\034\021\105\026\106\027\111" +
    "\032\223\033\067\034\102\035\104\036\077\037\074\040" +
    "\075\041\066\043\101\044\114\001\001\000\002\001\001" +
    "\000\010\046\162\047\152\050\160\001\001\000\002\001" +
    "\001\000\002\001\001\000\030\021\133\022\132\041\123" +
    "\042\230\043\101\044\114\051\135\052\136\053\127\054" +
    "\141\055\144\001\001\000\010\046\162\047\152\050\160" +
    "\001\001\000\034\021\105\026\106\027\111\032\232\033" +
    "\067\034\102\035\104\036\077\037\074\040\075\041\066" +
    "\043\101\044\114\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\016\006\070" +
    "\007\237\013\026\015\020\016\016\017\012\001\001\000" +
    "\002\001\001\000\030\021\133\022\132\041\123\042\241" +
    "\043\101\044\114\051\135\052\136\053\127\054\141\055" +
    "\144\001\001\000\010\046\162\047\152\050\160\001\001" +
    "\000\002\001\001\000\002\001\001\000\002\001\001" });

  /** Access to <code>reduce_goto</code> table. */
  public short[][] reduce_table() {return _reduce_table;}

  /** Instance of action encapsulation class. */
  protected CUP$EJayParser$actions action_obj;

  /** Action encapsulation object initializer. */
  protected void init_actions()
    {
      action_obj = new CUP$EJayParser$actions(this);
    }

  /** Invoke a user supplied parse action. */
  public java_cup.runtime.Symbol do_action(
    int                        act_num,
    java_cup.runtime.lr_parser parser,
    java.util.Stack            stack,
    int                        top)
    throws java.lang.Exception
  {
    /* call code in generated class */
    return action_obj.CUP$EJayParser$do_action(act_num, parser, stack, top);
  }
```

```
449
450   /** Indicates start state. */
451   public int start_state() {return 0;}
452   /** Indicates start production. */
453   public int start_production() {return 1;}
454
455   /** <code>EOF</code> Symbol index. */
456   public int EOF_sym() {return 0;}
457
458   /** <code>error</code> Symbol index. */
459   public int error_sym() {return 1;}
460
461
462
463
464     public void syntax_error(Symbol cur_token) {
465         report_error("Sytax error at line " + (cur_token.left+1) +
466             ", column " + cur_token.right, null);
467     }
468
469     public void initSymbolTable(int size) {
470         symtab = new SymbolTable(size);
471     }
472
473     public SymbolTable getSymbolTable() {
474         return symtab;
475     }
476
477     /** Reference to current symbol table.  This value changes as the program
478      * is parsed, such that symtab points to the symbol table for the scope in
479      * which the parse is currently active. */
480     public SymbolTable symtab;
481
482     /** Reference to the current function entry.  It is assigned each time a
483      * function entry is created. */
484     public FunctionEntry entry;
485
486     /** Reasonable size for a function symbol table. */
487     public final int FUNC_SYMTAB_SIZE = 25;
488
489     /** Reasonable size for a block symbol table. */
490     public final int BLOCK_SYMTAB_SIZE = 25;
491
492     /** Incrementing counter for declared structs.  This number is used to
493      * create a unique name for each struct, so it can be entered and
494      * subsequently retrieved from a symbol table. */
495     public int structNum;
496
497     /** Incrementing counter for nested blocks.  This number is used to create
498      * a unique name for each block, so it can be entered and subsequently
499      * retrieved from a symbol table. */
500     public int blockNum;
501
502     /** Enter the vars in the given tree node list in the current symtab.  The
503      * type of each var entry is the given type. */
504     public void enterVars(TypeNode type, TreeNodeList vars) {
505         TreeNode node;
506         TreeNodeList rest;
507         boolean done  = false;
508         for (node = vars.node, rest = vars.siblings; !done; ) {
509             symtab.enter(new VariableEntry(
510                 (String) (((LeafNode) node).value), type, false, 0,
511                     symtab.level));
512             if (rest == null) {
513                 done = true;
514             }
515             else {
516                 node = rest.node;
517                 rest = rest.siblings;
518             }
519         }
520     }
521
522     /** Enter an inner block by allocating a symtab for it, entering a
523      * unique-named BlockEntry for it in the current symtab, and descending
524      * into the new block symtab.  This function is used both for inner
525      * statement blocks and struct types; the String-valued blockType parameter
526      * is "block" or "struct" depending on the caller.
527      *
528      * NOTE: An acceptable solution to CSC 330 Assignment 3 does not need to
529      * generate the unique entry name. */
530     public void enterBlock(String blockType) {
531         SymbolTable blockTab = new SymbolTable(BLOCK_SYMTAB_SIZE);
532         BlockEntry blockEntry = new BlockEntry(blockTab);
533         blockEntry.name = blockType + Integer.toString(
534             blockType.equals("block") ? blockNum++ : structNum++);
535         symtab.enter(blockEntry);
536         blockTab.parent = symtab;
537         symtab = blockTab;
538     }
539
540
541 }
542
543 /** Cup generated class to encapsulate user supplied action code.*/
544 class CUP$EJayParser$actions {
545   private final EJayParser parser;
546
547   /** Constructor */
548   CUP$EJayParser$actions(EJayParser parser) {
549     this.parser = parser;
550   }
551
552   /** Method with the actual generated action code. */
553   public final java_cup.runtime.Symbol CUP$EJayParser$do_action(
554     int                        CUP$EJayParser$act_num,
555     java_cup.runtime.lr_parser CUP$EJayParser$parser,
556     java.util.Stack            CUP$EJayParser$stack,
557     int                        CUP$EJayParser$top)
558     throws java.lang.Exception
559     {
560       /* Symbol object for return from actions */
```

```
        java_cup.runtime.Symbol CUP$EJayParser$result;

        /* select the action based on the action number */
        switch (CUP$EJayParser$act_num)
          {
            /*. . . . . .*/
            case 99: // Boolean ::= FALSE
              {
                TreeNode RESULT = null;
                  int fleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int fright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  Object f = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new LeafNode(sym.IDENT, null, fleft, fright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(43/*Boolean*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 98: // Boolean ::= TRUE
              {
                TreeNode RESULT = null;
                  int tleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int tright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  Object t = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new LeafNode(sym.IDENT, null, tleft, tright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(43/*Boolean*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 97: // String ::= STRING_LIT
              {
                TreeNode RESULT = null;
                  int slleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int slright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  Object sl = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new LeafNode(sym.STRING_LIT, sl, slleft, slright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(42/*String*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 96: // Float ::= FLOATING_PT
              {
                TreeNode RESULT = null;
                  int fpleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int fpright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  Object fp = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new LeafNode(sym.FLOATING_PT, fp, fpleft, fpright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(41/*Float*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 95: // Integer ::= INTEGER
              {
                TreeNode RESULT = null;
                  int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  Object i = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new LeafNode(sym.INTEGER, i, ileft, iright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(16/*Integer*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 94: // Identifier ::= IDENT
              {
                TreeNode RESULT = null;
                  int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  Object i = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new LeafNode(sym.IDENT, i, ileft, iright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(15/*Identifier*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 93: // Literal ::= Boolean
              {
                TreeNode RESULT = null;
                  int bleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int bright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  TreeNode b = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = b;
                CUP$EJayParser$result = new java_cup.runtime.Symbol(40/*Literal*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 92: // Literal ::= String
              {
                TreeNode RESULT = null;
                  int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = s;
                CUP$EJayParser$result = new java_cup.runtime.Symbol(40/*Literal*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 91: // Literal ::= Float
              {
                TreeNode RESULT = null;
                  int fleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int fright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  TreeNode f = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = f;
                CUP$EJayParser$result = new java_cup.runtime.Symbol(40/*Literal*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;
```

```
673
674           /*. . . . . .*/
675           case 90: // Literal ::= Integer
676             {
677               TreeNode RESULT = null;
678                 int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
679                 int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
680                 TreeNode i = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
681                  RESULT = i;
682               CUP$EJayParser$result = new java_cup.runtime.Symbol(40/*Literal*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
683             }
684           return CUP$EJayParser$result;
685
686           /*. . . . . .*/
687           case 89: // Identifiers ::= Identifier COMMA Identifiers
688             {
689               TreeNodeList RESULT = null;
690                 int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
691                 int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
692                 LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
693                 int isleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
694                 int isright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
695                 TreeNodeList is = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
696                  RESULT = new TreeNodeList(i, is);
697               CUP$EJayParser$result = new java_cup.runtime.Symbol(10/*Identifiers*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
698             }
699           return CUP$EJayParser$result;
700
701           /*. . . . . .*/
702           case 88: // Identifiers ::= Identifier
703             {
704               TreeNodeList RESULT = null;
705                 int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
706                 int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
707                 LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
708                  RESULT = new TreeNodeList(i, null);
709               CUP$EJayParser$result = new java_cup.runtime.Symbol(10/*Identifiers*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
710             }
711           return CUP$EJayParser$result;
712
713           /*. . . . . .*/
714           case 87: // Expressions ::= Expression COMMA Expressions
715             {
716               TreeNodeList RESULT = null;
717                 int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
718                 int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
719                 TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
720                 int esleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
721                 int esright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
722                 TreeNodeList es = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
723                  RESULT = new TreeNodeList(e, es);
724               CUP$EJayParser$result = new java_cup.runtime.Symbol(35/*Expressions*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
725             }
726           return CUP$EJayParser$result;
727
728           /*. . . . . .*/
729           case 86: // Expressions ::= Expression
730             {
731               TreeNodeList RESULT = null;
732                 int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
733                 int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
734                 TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
735                 RESULT = new TreeNodeList(e, null);
736               CUP$EJayParser$result = new java_cup.runtime.Symbol(35/*Expressions*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
737             }
738           return CUP$EJayParser$result;
739
740           /*. . . . . .*/
741           case 85: // UnaryOp ::= NOT
742             {
743               TreeNode1 RESULT = null;
744                 int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
745                 int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
746                 Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
747                  RESULT = new TreeNode1(sym.NOT, null, opleft, opright);
748               CUP$EJayParser$result = new java_cup.runtime.Symbol(39/*UnaryOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
749             }
750           return CUP$EJayParser$result;
751
752           /*. . . . . .*/
753           case 84: // UnaryOp ::= MINUS
754             {
755               TreeNode1 RESULT = null;
756                 int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
757                 int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
758                 Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
759                  RESULT = new TreeNode1(sym.UNY_MINUS, null, opleft, opright);
760               CUP$EJayParser$result = new java_cup.runtime.Symbol(39/*UnaryOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
761             }
762           return CUP$EJayParser$result;
763
764           /*. . . . . .*/
765           case 83: // UnaryOp ::= PLUS
766             {
767               TreeNode1 RESULT = null;
768                 int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
769                 int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
770                 Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
771                  RESULT = new TreeNode1(sym.UNY_PLUS, null, opleft, opright);
772               CUP$EJayParser$result = new java_cup.runtime.Symbol(39/*UnaryOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
773             }
774           return CUP$EJayParser$result;
775
776           /*. . . . . .*/
777           case 82: // MultiplicativeOp ::= DIVIDE
778             {
779               TreeNode2 RESULT = null;
780                 int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
781                 int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
782                 Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
783                  RESULT = new TreeNode2(sym.DIVIDE, null, null,
784                            opleft, opright);
```

```
785             CUP$EJayParser$result = new java_cup.runtime.Symbol(38/*MultiplicativeOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
786           }
787         return CUP$EJayParser$result;
788
789         /*. . . . . .*/
790         case 81: // MultiplicativeOp ::= TIMES
791           {
792             TreeNode2 RESULT = null;
793               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
794               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
795               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
796                RESULT = new TreeNode2(sym.TIMES, null, null,
797                           opleft, opright);
798             CUP$EJayParser$result = new java_cup.runtime.Symbol(38/*MultiplicativeOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
799           }
800         return CUP$EJayParser$result;
801
802         /*. . . . . .*/
803         case 80: // AdditiveOp ::= MINUS
804           {
805             TreeNode2 RESULT = null;
806               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
807               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
808               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
809                RESULT = new TreeNode2(sym.MINUS, null, null,
810                           opleft, opright);
811             CUP$EJayParser$result = new java_cup.runtime.Symbol(37/*AdditiveOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
812           }
813         return CUP$EJayParser$result;
814
815         /*. . . . . .*/
816         case 79: // AdditiveOp ::= PLUS
817           {
818             TreeNode2 RESULT = null;
819               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
820               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
821               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
822                RESULT = new TreeNode2(sym.PLUS, null, null,
823                           opleft, opright);
824             CUP$EJayParser$result = new java_cup.runtime.Symbol(37/*AdditiveOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
825           }
826         return CUP$EJayParser$result;
827
828         /*. . . . . .*/
829         case 78: // RelationalOp ::= NOT_EQ
830           {
831             TreeNode2 RESULT = null;
832               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
833               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
834               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
835                RESULT = new TreeNode2(sym.NOT_EQ, null, null,
836                           opleft, opright);
837             CUP$EJayParser$result = new java_cup.runtime.Symbol(36/*RelationalOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
838           }
839         return CUP$EJayParser$result;
840
841         /*. . . . . .*/
842         case 77: // RelationalOp ::= EQ_EQ
843           {
844             TreeNode2 RESULT = null;
845               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
846               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
847               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
848                RESULT = new TreeNode2(sym.EQ_EQ, null, null,
849                           opleft, opright);
850             CUP$EJayParser$result = new java_cup.runtime.Symbol(36/*RelationalOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
851           }
852         return CUP$EJayParser$result;
853
854         /*. . . . . .*/
855         case 76: // RelationalOp ::= GTR_EQ
856           {
857             TreeNode2 RESULT = null;
858               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
859               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
860               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
861                RESULT = new TreeNode2(sym.GTR_EQ, null, null,
862                           opleft, opright);
863             CUP$EJayParser$result = new java_cup.runtime.Symbol(36/*RelationalOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
864           }
865         return CUP$EJayParser$result;
866
867         /*. . . . . .*/
868         case 75: // RelationalOp ::= GTR
869           {
870             TreeNode2 RESULT = null;
871               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
872               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
873               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
874                RESULT = new TreeNode2(sym.GTR, null, null,
875                           opleft, opright);
876             CUP$EJayParser$result = new java_cup.runtime.Symbol(36/*RelationalOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
877           }
878         return CUP$EJayParser$result;
879
880         /*. . . . . .*/
881         case 74: // RelationalOp ::= LESS_EQ
882           {
883             TreeNode2 RESULT = null;
884               int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
885               int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
886               Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
887                RESULT = new TreeNode2(sym.LESS_EQ, null, null,
888                           opleft, opright);
889             CUP$EJayParser$result = new java_cup.runtime.Symbol(36/*RelationalOp*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
890           }
891         return CUP$EJayParser$result;
892
893         /*. . . . . .*/
894         case 73: // RelationalOp ::= LESS
895           {
896             TreeNode2 RESULT = null;
```

```
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNode2(sym.LESS, null, null,
                        opleft, opright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(36/*RelationalOp*/,
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 72: // Expression ::= LEFT_PAREN Expression RT_PAREN
          {
            TreeNode RESULT = null;
              int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
               RESULT = e;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/,
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 71: // Expression ::= Literal
          {
            TreeNode RESULT = null;
              int lleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int lright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode l = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = l;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/,
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 70: // Expression ::= Designator
          {
            TreeNode RESULT = null;
              int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode d = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = d;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/,
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 69: // Expression ::= Identifier LEFT_PAREN Expressions RT_PAREN
          {
            TreeNode RESULT = null;
              int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
              int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
              LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
              int esleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int esright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNodeList es = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
               RESULT = new TreeNode2(sym.LEFT_PAREN, i, es,
                        ileft, iright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 68: // Expression ::= Identifier LEFT_PAREN RT_PAREN
          {
            TreeNode RESULT = null;
              int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
               RESULT = new TreeNode2(sym.LEFT_PAREN, i, null,
                        ileft, iright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 67: // Expression ::= UnaryOp Expression
          {
            TreeNode RESULT = null;
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode1 op = (TreeNode1)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int e1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int e1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode e1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = op; op.child = e1;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 66: // Expression ::= Expression MultiplicativeOp Expression
          {
            TreeNode RESULT = null;
              int e1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int e1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNode e1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode2 op = (TreeNode2)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int e2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int e2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode e2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = op; op.child1 = e1; op.child2 = e2;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 65: // Expression ::= Expression AdditiveOp Expression
          {
            TreeNode RESULT = null;
              int e1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
```

```
              int e1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNode e1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode2 op = (TreeNode2)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int e2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int e2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode e2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = op; op.child1 = e1; op.child2 = e2;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 64: // Expression ::= Expression RelationalOp Expression
          {
            TreeNode RESULT = null;
              int e1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int e1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNode e1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode2 op = (TreeNode2)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int e2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int e2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode e2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = op; op.child1 = e1; op.child2 = e2;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 63: // Expression ::= Expression AND Expression
          {
            TreeNode RESULT = null;
              int e1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int e1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNode e1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int e2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int e2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode e2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNode2(sym.AND, e1, e2,
                        opleft, opright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 62: // Expression ::= Expression OR Expression
          {
            TreeNode RESULT = null;
              int e1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int e1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNode e1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int opleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int opright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              Object op = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int e2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int e2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode e2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNode2(sym.OR, e1, e2, opleft, opright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(32/*Expression*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 61: // PrintStatement ::= PRINT Expressions SEMI
          {
            TreeNode RESULT = null;
              int pleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int pright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              Object p = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNodeList es = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
               RESULT = new TreeNode1(sym.PRINT, es,
                        pleft, pright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(30/*PrintStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 60: // ReturnStatement ::= RETURN Expression SEMI
          {
            TreeNode RESULT = null;
              int rleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int rright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              Object r = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
               RESULT = new TreeNode1(sym.RETURN, e,
                        rleft, rright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(29/*ReturnStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 59: // CallStatement ::= Identifier LEFT_PAREN Expressions RT_PAREN SEMI
          {
            TreeNode RESULT = null;
              int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
              int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
              LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
              int esleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int esright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNodeList es = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
               RESULT = new TreeNode2(sym.LEFT_PAREN, i, es,
                        ileft, iright);
```

```
    CUP$EJayParser$result = new java_cup.runtime.Symbol(28/*CallStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
return CUP$EJayParser$result;

/*. . . . . .*/
case 58: // CallStatement ::= Identifier LEFT_PAREN RT_PAREN SEMI
  {
    TreeNode RESULT = null;
      int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
      int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
      LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
       RESULT = new TreeNode2(sym.LEFT_PAREN, i, null,
                   ileft, iright);
    CUP$EJayParser$result = new java_cup.runtime.Symbol(28/*CallStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
return CUP$EJayParser$result;

/*. . . . . .*/
case 57: // WhileStatement ::= WHILE LEFT_PAREN Expression RT_PAREN Statement
  {
    TreeNode RESULT = null;
      int wleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
      int wright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
      Object w = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
      int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
      int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
      TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
      int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
      int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
      TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
       RESULT = new TreeNode2(sym.WHILE, e, s,
                   wleft, wright);
    CUP$EJayParser$result = new java_cup.runtime.Symbol(27/*WhileStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
return CUP$EJayParser$result;

/*. . . . . .*/
case 56: // IfStatement ::= IF LEFT_PAREN Expression RT_PAREN Statement ELSE Statement
  {
    TreeNode RESULT = null;
      int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).left;
      int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).right;
      Object i = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).value;
      int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
      int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
      TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
      int s1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
      int s1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
      TreeNode s1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
      int s2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
      int s2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
      TreeNode s2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
       RESULT = new TreeNode3(sym.IF, e, s1, s2,
                   ileft, iright);
    CUP$EJayParser$result = new java_cup.runtime.Symbol(26/*IfStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
    CUP$EJayParser$result = new java_cup.runtime.Symbol(28/*CallStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
return CUP$EJayParser$result;

/*. . . . . .*/
case 57: // WhileStatement ::= WHILE LEFT_PAREN Expression RT_PAREN Statement
  {
    TreeNode RESULT = null;
      int wleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
      int wright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
      Object w = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
      int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
      int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
      TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
      int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
      int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
      TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
       RESULT = new TreeNode2(sym.WHILE, e, s,
                   wleft, wright);
    CUP$EJayParser$result = new java_cup.runtime.Symbol(27/*WhileStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
return CUP$EJayParser$result;

/*. . . . . .*/
case 56: // IfStatement ::= IF LEFT_PAREN Expression RT_PAREN Statement ELSE Statement
  {
    TreeNode RESULT = null;
      int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).left;
      int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).right;
      Object i = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).value;
      int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
      int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
      TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
      int s1left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
      int s1right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
      TreeNode s1 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
      int s2left = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
      int s2right = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
      TreeNode s2 = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
       RESULT = new TreeNode3(sym.IF, e, s1, s2,
                   ileft, iright);
    CUP$EJayParser$result = new java_cup.runtime.Symbol(26/*IfStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-6)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
  }
            /*. . . . . .*/
            case 55: // IfStatement ::= IF LEFT_PAREN Expression RT_PAREN Statement
              {
                TreeNode RESULT = null;
                  int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
                  int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
                  Object i = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
                  int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
                  int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
                  TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
                  int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new TreeNode3(sym.IF, e, s, null,
                               ileft, iright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(26/*IfStatement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 54: // StructSelection ::= Designator DOT Identifier
              {
                TreeNode RESULT = null;
                  int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
                  int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
                  TreeNode d = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
                  int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                  int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                  LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
                   RESULT = new TreeNode2(sym.DOT, d, i,
                               dleft, iright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(34/*StructSelection*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 53: // ArraySelection ::= Designator LEFT_BRKT Expressions RT_BRKT
              {
                TreeNode RESULT = null;
                  int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
                  int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
                  TreeNode d = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
                  int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
                  int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
                  TreeNodeList es = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
                   RESULT = new TreeNode2(sym.LEFT_BRKT, d, es,
                               dleft, iright);
                CUP$EJayParser$result = new java_cup.runtime.Symbol(33/*ArraySelection*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              }
            return CUP$EJayParser$result;

            /*. . . . . .*/
            case 52: // Designator ::= Identifier
              {
```

```
        TreeNode RESULT = null;
          int ssleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int ssright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode ss = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = ss;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(31/*Designator*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 51: // Designator ::= ArraySelection
      {
        TreeNode RESULT = null;
          int asleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int asright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode as = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = as;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(31/*Designator*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 50: // Designator ::= Identifier
      {
        TreeNode RESULT = null;
          int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = i;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(31/*Designator*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 49: // Assignment ::= Designator EQ Expression SEMI
      {
        TreeNode RESULT = null;
          int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
          int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
          TreeNode d = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
          int eleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
          int eright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
          TreeNode e = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
           RESULT = new TreeNode2(sym.EQ, d, e, dleft, dright);
        CUP$EJayParser$result = new java_cup.runtime.Symbol(25/*Assignment*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 48: // Statement ::= PrintStatement
      {
        TreeNode RESULT = null;
          int psleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int psright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode ps = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = ps;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 47: // Statement ::= WhileStatement
      {
        TreeNode RESULT = null;
          int wsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int wsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode ws = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = ws;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 46: // Statement ::= IfStatement
      {
        TreeNode RESULT = null;
          int isleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int isright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode is = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = is;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 45: // Statement ::= Assignment
      {
        TreeNode RESULT = null;
          int csleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int csright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode cs = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = cs;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 44: // Statement ::= Block
      {
        TreeNode RESULT = null;
          int rsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int rsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode rs = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = rs;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 43: // Statement ::= BlankStatement
      {
        TreeNode RESULT = null;
          int bsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int bsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode bs = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = bs;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 42: // Statement ::= ReturnStatement
      {
        TreeNode RESULT = null;
          int rsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int rsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode rs = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = rs;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 41: // Statement ::= IfStatement
      {
        TreeNode RESULT = null;
          int isleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int isright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode is = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = is;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 40: // Statement ::= Assignment
      {
        TreeNode RESULT = null;
```

```
        int aleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int aright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        TreeNode a = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = a;
      CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, aleft, aright, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 42: // Statement ::= Block
    {
      TreeNode RESULT = null;
        int bleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int bright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        TreeNode b = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = b;
      CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, bleft, bright, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 41: // Statement ::= SEMI
    {
      TreeNode RESULT = null;
        int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        Object s = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = new LeafNode(sym.SEMI, null, sleft, sright);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(24/*Statement*/, sleft, sright, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 40: // Statements ::= Statement Statements
    {
      TreeNodeList RESULT = null;
        int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
        int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
        TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
        int ssleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int ssright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        TreeNodeList ss = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = new TreeNodeList(s, ss);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(23/*Statements*/, sleft, ssright, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 39: // Statements ::= Statement
    {
      TreeNodeList RESULT = null;
        int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = new TreeNodeList(s, null);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(23/*Statements*/, sleft, sright, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 38: // DataDeclarations ::= DataDeclarations DataDeclaration SEMI
    {
      TreeNodeList RESULT = null;
        int ddleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
        int ddright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
        TreeNode dd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
        int ddsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int ddsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        TreeNodeList dds = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = new TreeNodeList(dd, dds);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(5/*DataDeclarations*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 37: // DataDeclarations ::= DataDeclaration SEMI
    {
      TreeNodeList RESULT = null;
        int ddleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
        int ddright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
        TreeNode dd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
         RESULT = new TreeNodeList(dd, null);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(5/*DataDeclarations*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 36: // BlockHeader ::= LEFT_BRACE
    {
      TreeNode RESULT = null;
         parser.enterBlock("block");
      CUP$EJayParser$result = new java_cup.runtime.Symbol(21/*BlockHeader*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 35: // Block ::= BlockHeader DataDeclarations Statements RT_BRACE
    {
      TreeNode RESULT = null;
        int bleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
        int bright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
        TreeNode b = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
        int ddsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
        int ddsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
        TreeNodeList dds = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
        int ssleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
        int ssright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
        TreeNodeList ss = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
         RESULT = new TreeNode2(sym.LEFT_BRACE, dds, ss,
                                parser.symtab.getCurrentBlock());
                                parser.symtab = parser.symtab.ascend();
      CUP$EJayParser$result = new java_cup.runtime.Symbol(21/*BlockHeader*/, bleft, bright, RESULT);
    }
  return CUP$EJayParser$result;
```

```
            }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 34: // Block ::= LEFT_BRACE Statements RT_BRACE
          {
            TreeNode RESULT = null;
              int lleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int lright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              Object l = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int ssleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int ssright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNodeList ss = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
               RESULT = new TreeNode2(sym.LEFT_BRACE, null, ss,
                           lleft, lright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(20/*Block*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 33: // Block ::= LEFT_BRACE RT_BRACE
          {
            TreeNode RESULT = null;
              int lleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int lright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              Object l = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNode2(sym.LEFT_BRACE, null, null,
                  lleft, lright);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(20/*Block*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 32: // FormalDeclaration ::= REF Type Identifier
          {
            TreeNode RESULT = null;
              int rleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int rright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              Object r = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int tleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int tright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TypeNode t = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNode3(sym.COMMA, t, i,
                           new LeafNode(sym.REF, null, rleft, rright),
                               tleft, tright);
                      parser.symtab.enter(new VariableEntry(
                          (String) i.value, t, true, 0,
                               parser.symtab.level));
            CUP$EJayParser$result = new java_cup.runtime.Symbol(22/*FormalDeclaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 31: // FormalDeclaration ::= Type Identifier
          {
            TreeNode RESULT = null;
              int tleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int tright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TypeNode t = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNode3(sym.COMMA, t, i, null,
                   tleft, tright);
                      parser.symtab.enter(new VariableEntry(
                          (String) i.value, t, false, 0,
                               parser.symtab.level));
            CUP$EJayParser$result = new java_cup.runtime.Symbol(22/*FormalDeclaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 30: // FormalDeclaration ::=
          {
            TreeNode RESULT = null;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(22/*FormalDeclaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 29: // FormalDeclarations ::= FormalDeclarations COMMA FormalDeclaration
          {
            TreeNodeList RESULT = null;
              int fdleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
              int fdright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
              TreeNode fd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int fsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int fsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNodeList fs = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNodeList(fd, fs);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(19/*Formals*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 28: // Formals ::= FormalDeclaration
          {
            TreeNodeList RESULT = null;
              int fdleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
              int fdright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
              TreeNode fd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNodeList(fd, null);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(19/*Formals*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }
        return CUP$EJayParser$result;

        /*. . . . . .*/
        case 27: // FunctionBlock ::= LEFT_BRACE DataDeclarations Statements RT_BRACE
```

```
    {
      TreeNode RESULT = null;
        int lleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
        int lright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
        Object l = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
        int ddsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
        int ddsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
        TreeNodeList dds = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
        int ssleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
        int ssright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
        TreeNodeList ss = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
         RESULT = new TreeNode2(sym.LEFT_BRACE, dds, ss,
                  lleft, lright);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(8/*FunctionBlock*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 26: // FunctionBlock ::= LEFT_BRACE Statements RT_BRACE
    {
      TreeNode RESULT = null;
        int lleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
        int lright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
        Object l = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
        int ssleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
        int ssright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
        TreeNodeList ss = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
         RESULT = new TreeNode2(sym.LEFT_BRACE, null, ss,
                  lleft, lright);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(8/*FunctionBlock*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 25: // FunctionBlock ::= LEFT_BRACE RT_BRACE
    {
      TreeNode RESULT = null;
        int lleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
        int lright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
        Object l = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
         RESULT = new TreeNode2(sym.LEFT_BRACE, null, null,
                  lleft, lright);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(8/*FunctionBlock*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 24: // FunctionHeader ::= Type Identifier
    {
      TreeNode4 RESULT = null;
        int tleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1).left;
        int tright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1).right;
        TypeNode t = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
        int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        LeafNode i = (LeafNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = new TreeNode4(sym.LEFT_PAREN,
                  t, i, null, null, tleft, tright);
      parser.symtab = parser.symtab.newLevel(parser.entry =
        new FunctionEntry((String) i.value, t, null,
        null, null), parser.FUNC_SYMTAB_SIZE);
      CUP$EJayParser$result = new java_cup.runtime.Symbol(7/*FunctionHeader*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
    }
  return CUP$EJayParser$result;

  /*. . . . . .*/
  case 23: // FunctionDeclaration ::= FunctionHeader LEFT_PAREN Formals RT_PAREN FunctionBlock
    {
      TreeNode RESULT = null;
        int fhleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left;
        int fhright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).right;
        TreeNode4 fh = (TreeNode4)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).value;
        int fsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
        int fsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
        TreeNodeList fs = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
        int bleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
        int bright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        TreeNode b = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
         RESULT = fh; fh.child3 = fs; fh.child4 = b;
        parser.entry.formals = fs;
        parser.entry.body = b;
        parser.symtab = parser.symtab.ascend();
        CUP$EJayParser$result = new java_cup.runtime.Symbol(6/*FunctionDeclaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-4)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;
    /*. . . . . .*/
    case 22: // Field ::= DataDeclaration
      {
        TreeNode RESULT = null;
          int ddleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int ddright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNode dd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = dd;
        CUP$EJayParser$result = new java_cup.runtime.Symbol(18/*Field*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;

    /*. . . . . .*/
    case 21: // Fields ::= Field SEMI Fields
      {
        TreeNodeList RESULT = null;
          int fleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;
          int fright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;
          TreeNode f = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
          int fsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          int fsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
          TreeNodeList fs = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
           RESULT = new TreeNodeList(f, fs);
        CUP$EJayParser$result = new java_cup.runtime.Symbol(17/*Fields*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
      }
    return CUP$EJayParser$result;
```

```
                                                                        {
                                                                          TreeNodeList RESULT = null;
                                                                            int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                                                                            int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
                                                                            TreeNode i = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
        /*. . . . . .*/                                                     RESULT = new TreeNodeList(i, null);
        case 20: // Fields ::= Field SEMI                                   CUP$EJayParser$result = new java_cup.runtime.Symbol(14/*Dimensions*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          {                                                               return CUP$EJayParser$result;
            TreeNodeList RESULT = null;                                   /*. . . . . .*/
              int fleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int fright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode f = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
               RESULT = new TreeNodeList(f, null);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(17/*Fields*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }                                                               /*. . . . . .*/
        return CUP$EJayParser$result;                                     case 15: // Dimensions ::=
                                                                            {
                                                                              TreeNodeList RESULT = null;
        /*. . . . . .*/
        case 19: // StructHeader ::= STRUCT                                  CUP$EJayParser$result = new java_cup.runtime.Symbol(14/*Dimensions*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          {                                                               }
            TreeNode RESULT = null;                                     return CUP$EJayParser$result;
               parser.enterBlock("struct");                             CUP$EJayParser$result = new java_cup.runtime.Symbol(14/*Dimensions*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
            CUP$EJayParser$result = new java_cup.runtime.Symbol(13/*StructHeader*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }                                                               /*. . . . . .*/
        return CUP$EJayParser$result;                                     case 14: // ArrayType ::= Type LEFT_BRKT Dimensions RT_BRKT
                                                                            {
        /*. . . . . .*/                                                     TypeNode RESULT = null;
        case 18: // StructType ::= StructHeader LEFT_BRACE Fields RT_BRACE   int tleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left;
          {                                                                 int tright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).right;
            TypeNode RESULT = null;                                         TypeNode t = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).value;
              int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;             int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
              int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;           int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
              TreeNode s = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;  TreeNodeList d = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
              int fsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;          RESULT = new TypeNode(
              int fsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;            sym.LEFT_BRKT, t, ds, tleft, tright);
              TreeNodeList fs = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TypeNode(sym.LEFT_BRACE, fs, sleft, sright);    CUP$EJayParser$result = new java_cup.runtime.Symbol(11/*ArrayType*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
                    RESULT.symtab = parser.symtab;                        }
                    parser.symtab = parser.symtab.ascend();             return CUP$EJayParser$result;
            CUP$EJayParser$result = new java_cup.runtime.Symbol(12/*StructType*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-3)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }                                                               /*. . . . . .*/
        return CUP$EJayParser$result;                                     case 13: // Type ::= StructType
                                                                            {
        /*. . . . . .*/                                                     TypeNode RESULT = null;
        case 17: // Dimensions ::= Integer COMMA Dimensions                   int stleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
          {                                                                 int stright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
            TreeNodeList RESULT = null;                                     TypeNode st = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
              int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left;              RESULT = st;
              int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).right;         CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
              TreeNode i = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).value;
              int dsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;       return CUP$EJayParser$result;
              int dsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;      /*. . . . . .*/
              TreeNodeList ds = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
               RESULT = new TreeNodeList(i, ds);                            {
            CUP$EJayParser$result = new java_cup.runtime.Symbol(14/*Dimensions*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-2)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
          }                                                                 TypeNode RESULT = null;
        return CUP$EJayParser$result;                                       int atleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
                                                                            int atright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
        /*. . . . . .*/                                                     TypeNode at = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
        case 16: // Dimensions ::= Integer                                   RESULT = at;
                                                                          CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
```

```
1793            }
1794          return CUP$EJayParser$result;
1795
1796          /*. . . . . .*/
1797          case 11: // Type ::= VOID
1798            {
1799              TypeNode RESULT = null;
1800                int vleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1801                int vright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1802                Object v = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1803                 RESULT = new TypeNode(sym.VOID, vleft, vright);
1804              CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1805            }
1806          return CUP$EJayParser$result;
1807
1808          /*. . . . . .*/
1809          case 10: // Type ::= BOOLEAN
1810            {
1811              TypeNode RESULT = null;
1812                int bleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1813                int bright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1814                Object b = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1815                 RESULT = new TypeNode(sym.BOOLEAN, bleft, bright);
1816              CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1817            }
1818          return CUP$EJayParser$result;
1819
1820          /*. . . . . .*/
1821          case 9: // Type ::= STRING
1822            {
1823              TypeNode RESULT = null;
1824                int sleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1825                int sright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1826                Object s = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1827                 RESULT = new TypeNode(sym.STRING, sleft, sright);
1828              CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1829            }
1830          return CUP$EJayParser$result;
1831
1832          /*. . . . . .*/
1833          case 8: // Type ::= FLOAT
1834            {
1835              TypeNode RESULT = null;
1836                int fleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1837                int fright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1838                Object f = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1839                 RESULT = new TypeNode(sym.FLOAT, fleft, fright);
1840              CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1841            }
1842          return CUP$EJayParser$result;
1843
1844          /*. . . . . .*/
1845          case 7: // Type ::= INT
1846            {
1847              TypeNode RESULT = null;
1848                int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1849                int iright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1850                Object i = (Object)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1851                 RESULT = new TypeNode(sym.INT, ileft, iright);
1852              CUP$EJayParser$result = new java_cup.runtime.Symbol(9/*Type*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1853            }
1854          return CUP$EJayParser$result;
1855
1856          /*. . . . . .*/
1857          case 6: // DataDeclaration ::= Type Identifiers
1858            {
1859              TreeNode RESULT = null;
1860                int tleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
1861                int tright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
1862                TypeNode t = (TypeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
1863                int isleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1864                int isright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1865                TreeNodeList is = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1866                 RESULT = new TreeNode2(
1867                                sym.SEMI, t, is, tleft, tright);
1868                 parser.enterVars(t, is);
1869              CUP$EJayParser$result = new java_cup.runtime.Symbol(4/*DataDeclaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1870            }
1871          return CUP$EJayParser$result;
1872
1873          /*. . . . . .*/
1874          case 5: // Declaration ::= FunctionDeclaration
1875            {
1876              TreeNode RESULT = null;
1877                int fdleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1878                int fdright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1879                TreeNode fd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1880                 RESULT = fd;
1881              CUP$EJayParser$result = new java_cup.runtime.Symbol(3/*Declaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1882            }
1883          return CUP$EJayParser$result;
1884
1885          /*. . . . . .*/
1886          case 4: // Declaration ::= DataDeclaration SEMI
1887            {
1888              TreeNode RESULT = null;
1889                int ddleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
1890                int ddright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
1891                TreeNode dd = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
1892                 RESULT = dd;
1893              CUP$EJayParser$result = new java_cup.runtime.Symbol(3/*Declaration*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1894            }
1895          return CUP$EJayParser$result;
1896
1897          /*. . . . . .*/
1898          case 3: // Declarations ::= Declaration Declarations
1899            {
1900              TreeNodeList RESULT = null;
1901                int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
1902                int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
1903                TreeNode d = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
1904                int ileft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
```

```
1905                int dsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1906                TreeNodeList ds = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1907                 RESULT = new TreeNodeList(d, ds);
1908              CUP$EJayParser$result = new java_cup.runtime.Symbol(2/*Declarations*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1909            }
1910          return CUP$EJayParser$result;
1911
1912          /*. . . . . .*/
1913          case 2: // Declarations ::= Declaration
1914            {
1915              TreeNodeList RESULT = null;
1916                int dleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1917                int dright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1918                TreeNode d = (TreeNode)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1919                 RESULT = new TreeNodeList(d, null);
1920              CUP$EJayParser$result = new java_cup.runtime.Symbol(2/*Declarations*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1921            }
1922          return CUP$EJayParser$result;
1923
1924          /*. . . . . .*/
1925          case 1: // $START ::= Program EOF
1926            {
1927              Object RESULT = null;
1928                int start_valleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left;
1929                int start_valright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).right;
1930                TreeNodeList start_val = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).value;
1931                RESULT = start_val;
1932              CUP$EJayParser$result = new java_cup.runtime.Symbol(0/*$START*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-1)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1933            }
1934          /* ACCEPT */
1935          CUP$EJayParser$parser.done_parsing();
1936          return CUP$EJayParser$result;
1937
1938          /*. . . . . .*/
1939          case 0: // Program ::= Declarations
1940            {
1941              TreeNodeList RESULT = null;
1942                int dsleft = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left;
1943                int dsright = ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right;
1944                TreeNodeList ds = (TreeNodeList)((java_cup.runtime.Symbol) CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).value;
1945                 RESULT = ds;
1946              CUP$EJayParser$result = new java_cup.runtime.Symbol(1/*Program*/, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).left, ((java_cup.runtime.Symbol)CUP$EJayParser$stack.elementAt(CUP$EJayParser$top-0)).right, RESULT);
1947            }
1948          return CUP$EJayParser$result;
1949
1950          /* . . . . . .*/
1951          default:
1952            throw new Exception(
1953               "Invalid action number found in internal parse table");
1954
1955        }
1956    }
1957 }
1958
```