```
//----------------------------------------------------
// The following code was generated by CUP v0.11b 20160615 (GIT 4ac7450)
//----------------------------------------------------

package analizadores;

import java_cup.runtime.*;
import java.util.LinkedList;
import arbol.*;
import excepciones.Errores;
import java_cup.runtime.XMLElement;

/** CUP v0.11b 20160615 (GIT 4ac7450) generated parser.
  */
@SuppressWarnings({"rawtypes"})
public class Parser extends java_cup.runtime.lr_parser {

 public final Class getSymbolContainer() {
    return sym.class;
}

  /** Default constructor. */
  @Deprecated
  public Parser() {super();}

  /** Constructor which sets the default scanner. */
  @Deprecated
  public Parser(java_cup.runtime.Scanner s) {super(s);}

  /** Constructor which sets the default scanner. */
  public Parser(java_cup.runtime.Scanner s, java_cup.runtime.SymbolFactory sf) {super(s,sf);}

  /** Production table. */
  protected static final short _production_table[][] =
    unpackFromStrings(new String[] {
    "\000\200\000\002\002\004\000\002\002\003\000\002\003" +
    "\004\000\002\003\003\000\002\004\003\000\002\004\003" +
    "\000\002\004\003\000\002\004\003\000\002\004\003\000" +
    "\002\004\003\000\002\004\003\000\002\004\003\000\002" +
    "\004\003\000\002\004\003\000\002\004\003\000\002\004" +
    "\003\000\002\004\003\000\002\004\003\000\002\004\003" +
    "\000\002\004\004\000\002\004\004\000\002\016\007\000" +
    "\002\012\012\000\002\012\012\000\002\017\005\000\002" +
    "\017\003\000\002\017\002\000\002\013\004\000\002\005" +
    "\004\000\002\005\003\000\002\006\003\000\002\006\003" +
    "\000\002\006\003\000\002\006\003\000\002\006\003\000" +
    "\002\006\003\000\002\006\003\000\002\006\003\000\002" +
    "\006\003\000\002\006\003\000\002\006\003\000\002\006" +
    "\003\000\002\006\003\000\002\006\003\000\002\006\004" +
    "\000\002\006\004\000\002\007\011\000\002\007\007\000" +
    "\002\011\003\000\002\011\003\000\002\011\003\000\002" +
    "\011\003\000\002\011\003\000\002\010\006\000\002\010" +
    "\005\000\002\010\005\000\002\034\015\000\002\034\017" +
    "\000\002\040\005\000\002\040\003\000\002\035\011\000" +
    "\002\035\014\000\002\036\015\000\002\037\010\000\002" +
    "\037\010\000\002\021\007\000\002\021\007\000\002\022" +
    "\011\000\002\022\015\000\002\022\013\000\002\023\011" +
    "\000\002\033\004\000\002\033\003\000\002\024\006\000" +
    "\002\024\005\000\002\025\011\000\002\026\013\000\002" +
    "\032\005\000\002\032\004\000\002\032\004\000\002\027" +
    "\014\000\002\030\004\000\002\031\004\000\002\015\005" +
    "\000\002\015\004\000\002\041\004\000\002\041\006\000" +
    "\002\041\006\000\002\041\006\000\002\041\006\000\002" +
    "\041\006\000\002\041\005\000\002\041\005\000\002\041" +
    "\005\000\002\041\005\000\002\041\005\000\002\041\005" +
    "\000\002\041\005\000\002\041\005\000\002\041\005\000" +
    "\002\041\005\000\002\041\005\000\002\041\005\000\002" +
    "\041\005\000\002\041\005\000\002\041\005\000\002\041" +
    "\004\000\002\041\005\000\002\041\003\000\002\041\003" +
    "\000\002\041\003\000\002\041\003\000\002\041\003\000" +
    "\002\041\003\000\002\041\003\000\002\041\005\000\002" +
    "\041\006\000\002\041\011\000\002\041\006\000\002\041" +
    "\006\000\002\041\006\000\002\041\010\000\002\041\010" +
    "\000\002\041\003\000\002\014\006\000\002\020\005\000" +
    "\002\020\003\000\002\020\002" });

   /** Access to production table. */
   public short[][] production_table() {return _production_table;}

   /** Parse-action table. */
   protected static final short[][] _action_table =
    unpackFromStrings(new String[] {
    "\000\u013b\000\050\004\012\005\010\006\016\007\006\010" +
    "\045\011\030\012\051\013\022\015\032\020\040\021\053" +
    "\022\014\023\026\024\011\026\037\027\052\030\013\036" +
    "\041\100\025\001\002\000\052\002\ufff2\004\ufff2\005\ufff2" +
    "\006\ufff2\007\ufff2\010\ufff2\011\ufff2\012\ufff2\013\ufff2\015" +
    "\ufff2\020\ufff2\021\ufff2\022\ufff2\023\ufff2\024\ufff2\026\ufff2" +
    "\027\ufff2\030\ufff2\036\ufff2\100\ufff2\001\002\000\004\073" +
    "\u013d\001\002\000\014\057\uffce\071\uffce\073\uffce\077\uffce" +
    "\100\uffce\001\002\000\052\002\ufff7\004\ufff7\005\ufff7\006" +
    "\ufff7\007\ufff7\010\ufff7\011\ufff7\012\ufff7\013\ufff7\015\ufff7" +
    "\020\ufff7\021\ufff7\022\ufff7\023\ufff7\024\ufff7\026\ufff7\027" +
    "\ufff7\030\ufff7\036\ufff7\100\ufff7\001\002\000\014\057\uffd0" +
    "\071\uffd0\073\uffd0\077\uffd0\100\uffd0\001\002\000\004\073" +
    "\u013c\001\002\000\014\057\uffd1\071\uffd1\073\uffd1\077\uffd1" +
    "\100\uffd1\001\002\000\004\065\u0138\001\002\000\004\065" +
    "\u0129\001\002\000\052\002\ufff5\004\ufff5\005\ufff5\006\ufff5" +
    "\007\ufff5\010\ufff5\011\ufff5\012\ufff5\013\ufff5\015\ufff5\020" +
    "\ufff5\021\ufff5\022\ufff5\023\ufff5\024\ufff5\026\ufff5\027\ufff5" +
    "\030\ufff5\036\ufff5\100\ufff5\001\002\000\014\057\uffcf\071" +
    "\uffcf\073\uffcf\077\uffcf\100\uffcf\001\002\000\052\002\uffef" +
    "\004\uffef\005\uffef\006\uffef\007\uffef\010\uffef\011\uffef\012" +
    "\uffef\013\uffef\015\uffef\020\uffef\021\uffef\022\uffef\023\uffef" +
    "\024\uffef\026\uffef\027\uffef\030\uffef\036\uffef\100\uffef\001" +
    "\002\000\052\002\000\004\012\005\010\006\016\007\006" +
    "\010\045\011\030\012\051\013\022\015\032\020\040\021" +
    "\053\022\014\023\026\024\011\026\037\027\052\030\013" +
    "\036\041\100\025\001\002\000\052\002\ufff6\004\ufff6\005" +
    "\ufff6\006\ufff6\007\ufff6\010\ufff6\011\ufff6\012\ufff6\013\ufff6" +
    "\015\ufff6\020\ufff6\021\ufff6\022\ufff6\023\ufff6\024\ufff6\026" +
    "\ufff6\027\ufff6\030\ufff6\036\ufff6\100\ufff6\001\002\000\004" +
    "\065\u011d\001\002\000\052\002\ufff4\004\ufff4\005\ufff4\006" +
    "\ufff4\007\ufff4\010\ufff4\011\ufff4\012\ufff4\013\ufff4\015\ufff4" +
    "\020\ufff4\021\ufff4\022\ufff4\023\ufff4\024\ufff4\026\ufff4\027" +
    "\ufff4\030\ufff4\036\ufff4\100\ufff4\001\002\000\052\002\ufff1" +
    "\004\ufff1\005\ufff1\006\ufff1\007\ufff1\010\ufff1\011\ufff1\012" +
    "\ufff1\013\ufff1\015\ufff1\020\ufff1\021\ufff1\022\ufff1\023\ufff1" +
    "\024\ufff1\026\ufff1\027\ufff1\030\ufff1\036\ufff1\100\ufff1\001" +
    "\002\000\016\051\u0103\052\u0101\065\167\071\u0104\076\u0102" +
    "\077\u0105\001\002\000\004\073\u0100\001\002\000\052\002" +
    "\ufffb\004\ufffb\005\ufffb\006\ufffb\007\ufffb\010\ufffb\011\ufffb" +
    "\012\ufffb\013\ufffb\015\ufffb\020\ufffb\021\ufffb\022\ufffb\023" +
    "\ufffb\024\ufffb\026\ufffb\027\ufffb\030\ufffb\036\ufffb\100\ufffb" +
    "\001\002\000\004\100\371\001\002\000\052\002\ufffa\004" +
    "\ufffa\005\ufffa\006\ufffa\007\ufffa\010\ufffa\011\ufffa\012\ufffa" +
    "\013\ufffa\015\ufffa\020\ufffa\021\ufffa\022\ufffa\023\ufffa\024" +
    "\ufffa\026\ufffa\027\ufffa\030\ufffa\036\ufffa\100\ufffa\001\002" +
    "\000\004\065\352\001\002\000\004\073\351\001\002\000" +
    "\052\002\ufff0\004\ufff0\005\ufff0\006\ufff0\007\ufff0\010\ufff0" +
    "\011\ufff0\012\ufff0\013\ufff0\015\ufff0\020\ufff0\021\ufff0\022" +
    "\ufff0\023\ufff0\024\ufff0\026\ufff0\027\ufff0\030\ufff0\036\ufff0" +
    "\100\ufff0\001\002\000\004\100\335\001\002\000\052\002" +
    "\ufffc\004\ufffc\005\ufffc\006\ufffc\007\ufffc\010\ufffc\011\ufffc" +
    "\012\ufffc\013\ufffc\015\ufffc\020\ufffc\021\ufffc\022\ufffc\023" +
    "\ufffc\024\ufffc\026\ufffc\027\ufffc\030\ufffc\036\ufffc\100\ufffc" +
    "\001\002\000\004\100\331\001\002\000\004\065\323\001" +
    "\002\000\004\055\311\001\002\000\052\002\ufff9\004\ufff9" +
    "\005\ufff9\006\ufff9\007\ufff9\010\ufff9\011\ufff9\012\ufff9\013" +
    "\ufff9\015\ufff9\020\ufff9\021\ufff9\022\ufff9\023\ufff9\024\ufff9" +
    "\026\ufff9\027\ufff9\030\ufff9\036\ufff9\100\ufff9\001\002\000" +
    "\004\002\310\001\002\000\052\002\ufffe\004\ufffe\005\ufffe" +
    "\006\ufffe\007\ufffe\010\ufffe\011\ufffe\012\ufffe\013\ufffe\015" +
    "\ufffe\020\ufffe\021\ufffe\022\ufffe\023\ufffe\024\ufffe\026\ufffe" +
    "\027\ufffe\030\ufffe\036\ufffe\100\ufffe\001\002\000\014\057" +
    "\uffcd\071\uffcd\073\uffcd\077\uffcd\100\uffcd\001\002\000\052" +
    "\002\ufff3\004\ufff3\005\ufff3\006\ufff3\007\ufff3\010\ufff3\011" +
    "\ufff3\012\ufff3\013\ufff3\015\ufff3\020\ufff3\021\ufff3\022\ufff3" +
    "\023\ufff3\024\ufff3\026\ufff3\027\ufff3\030\ufff3\036\ufff3\100" +
    "\ufff3\001\002\000\052\002\ufffd\004\ufffd\005\ufffd\006\ufffd" +
    "\007\ufffd\010\ufffd\011\ufffd\012\ufffd\013\ufffd\015\ufffd\020" +
    "\ufffd\021\ufffd\022\ufffd\023\ufffd\024\ufffd\026\ufffd\027\ufffd" +
    "\030\ufffd\036\ufffd\100\ufffd\001\002\000\052\002\ufff8\004" +
    "\ufff8\005\ufff8\006\ufff8\007\ufff8\010\ufff8\011\ufff8\012\ufff8" +
    "\013\ufff8\015\ufff8\020\ufff8\021\ufff8\022\ufff8\023\ufff8\024" +
    "\ufff8\026\ufff8\027\ufff8\030\ufff8\036\ufff8\100\ufff8\001\002" +
    "\000\004\100\263\001\002\000\004\065\257\001\002\000" +
    "\004\067\054\001\002\000\034\012\051\013\022\015\032" +
    "\020\040\021\053\022\014\023\026\024\011\025\061\027" +
    "\052\030\013\036\041\100\025\001\002\000\036\012\051" +
    "\013\022\015\032\020\040\021\053\022\014\023\026\024" +
    "\011\025\061\027\052\030\013\036\041\070\251\100\025" +
    "\001\002\000\042\012\uffdc\013\uffdc\015\uffdc\016\uffdc\017" +
    "\uffdc\020\uffdc\021\uffdc\022\uffdc\023\uffdc\024\uffdc\025\uffdc" +
    "\027\uffdc\030\uffdc\036\uffdc\070\uffdc\100\uffdc\001\002\000" +
    "\042\012\uffd8\013\uffd8\015\uffd8\016\uffd8\017\uffd8\020\uffd8" +
    "\021\uffd8\022\uffd8\023\uffd8\024\uffd8\025\uffd8\027\uffd8\030" +
    "\uffd8\036\uffd8\070\uffd8\100\uffd8\001\002\000\042\012\uffdd" +
    "\013\uffdd\015\uffdd\016\uffdd\017\uffdd\020\uffdd\021\uffdd\022" +
    "\uffdd\023\uffdd\024\uffdd\025\uffdd\027\uffdd\030\uffdd\036\uffdd" +
    "\070\uffdd\100\uffdd\001\002\000\040\031\105\032\117\033" +
    "\102\041\107\042\115\044\106\063\122\065\103\071\113" +
    "\073\111\100\116\101\110\102\104\103\120\104\112\001" +
    "\002\000\042\012\uffe4\013\uffe4\015\uffe4\016\uffe4\017\uffe4" +
    "\020\uffe4\021\uffe4\022\uffe4\023\uffe4\024\uffe4\025\uffe4\027" +
    "\uffe4\030\uffe4\036\uffe4\070\uffe4\100\uffe4\001\002\000\042" +
    "\012\uffd7\013\uffd7\015\uffd7\016\uffd7\017\uffd7\020\uffd7\021" +
    "\uffd7\022\uffd7\023\uffd7\024\uffd7\025\uffd7\027\uffd7\030\uffd7" +
    "\036\uffd7\070\uffd7\100\uffd7\001\002\000\042\012\uffe0\013" +
    "\uffe0\015\uffe0\016\uffe0\017\uffe0\020\uffe0\021\uffe0\022\uffe0" +
    "\023\uffe0\024\uffe0\025\uffe0\027\uffe0\030\uffe0\036\uffe0\070" +
    "\uffe0\100\uffe0\001\002\000\042\012\uffe2\013\uffe2\015\uffe2" +
    "\016\uffe2\017\uffe2\020\uffe2\021\uffe2\022\uffe2\023\uffe2\024" +
    "\uffe2\025\uffe2\027\uffe2\030\uffe2\036\uffe2\070\uffe2\100\uffe2" +
    "\001\002\000\042\012\uffe3\013\uffe3\015\uffe3\016\uffe3\017" +
    "\uffe3\020\uffe3\021\uffe3\022\uffe3\023\uffe3\024\uffe3\025\uffe3" +
    "\027\uffe3\030\uffe3\036\uffe3\070\uffe3\100\uffe3\001\002\000" +
    "\042\012\uffd9\013\uffd9\015\uffd9\016\uffd9\017\uffd9\020\uffd9" +
    "\021\uffd9\022\uffd9\023\uffd9\024\uffd9\025\uffd9\027\uffd9\030" +
    "\uffd9\036\uffd9\070\uffd9\100\uffd9\001\002\000\004\073\101" +
    "\001\002\000\004\073\100\001\002\000\042\012\uffde\013" +
    "\uffde\015\uffde\016\uffde\017\uffde\020\uffde\021\uffde\022\uffde" +
    "\023\uffde\024\uffde\025\uffde\027\uffde\030\uffde\036\uffde\070" +
    "\uffde\100\uffde\001\002\000\042\012\uffe1\013\uffe1\015\uffe1" +
    "\016\uffe1\017\uffe1\020\uffe1\021\uffe1\022\uffe1\023\uffe1\024" +
    "\uffe1\025\uffe1\027\uffe1\030\uffe1\036\uffe1\070\uffe1\100\uffe1" +
    "\001\002\000\042\012\uffdf\013\uffdf\015\uffdf\016\uffdf\017" +
    "\uffdf\020\uffdf\021\uffdf\022\uffdf\023\uffdf\024\uffdf\025\uffdf" +
    "\027\uffdf\030\uffdf\036\uffdf\070\uffdf\100\uffdf\001\002\000" +
    "\042\012\uffdb\013\uffdb\015\uffdb\016\uffdb\017\uffdb\020\uffdb" +
    "\021\uffdb\022\uffdb\023\uffdb\024\uffdb\025\uffdb\027\uffdb\030" +
    "\uffdb\036\uffdb\070\uffdb\100\uffdb\001\002\000\042\012\uffda" +
    "\013\uffda\015\uffda\016\uffda\017\uffda\020\uffda\021\uffda\022" +
    "\uffda\023\uffda\024\uffda\025\uffda\027\uffda\030\uffda\036\uffda" +
    "\070\uffda\100\uffda\001\002\000\042\012\uffd6\013\uffd6\015" +
    "\uffd6\016\uffd6\017\uffd6\020\uffd6\021\uffd6\022\uffd6\023\uffd6" +
    "\024\uffd6\025\uffd6\027\uffd6\030\uffd6\036\uffd6\070\uffd6\100" +
    "\uffd6\001\002\000\042\012\uffd4\013\uffd4\015\uffd4\016\uffd4" +
    "\017\uffd4\020\uffd4\021\uffd4\022\uffd4\023\uffd4\024\uffd4\025" +
    "\uffd4\027\uffd4\030\uffd4\036\uffd4\070\uffd4\100\uffd4\001\002" +
    "\000\042\012\uffd5\013\uffd5\015\uffd5\016\uffd5\017\uffd5\020" +
    "\uffd5\021\uffd5\022\uffd5\023\uffd5\024\uffd5\025\uffd5\027\uffd5" +
    "\030\uffd5\036\uffd5\070\uffd5\100\uffd5\001\002\000\004\065" +
    "\245\001\002\000\050\004\224\005\231\006\230\007\225" +
    "\010\227\031\105\032\117\033\102\041\107\042\115\044" +
    "\106\063\122\065\103\071\113\100\116\101\110\102\104" +
    "\103\120\104\112\001\002\000\052\043\uff93\044\uff93\045" +
    "\uff93\046\uff93\047\uff93\050\uff93\053\uff93\054\uff93\055\uff93" +
    "\056\uff93\057\uff93\060\uff93\061\uff93\062\uff93\064\uff93\066" +
    "\uff93\072\uff93\073\uff93\074\uff93\075\uff93\001\002\000\004" +
    "\065\221\001\002\000\036\031\105\032\117\033\102\041" +
    "\107\042\115\044\106\063\122\065\103\071\113\100\116" +
    "\101\110\102\104\103\120\104\112\001\002\000\052\043" +
    "\uff90\044\uff90\045\uff90\046\uff90\047\uff90\050\uff90\053\uff90" +
    "\054\uff90\055\uff90\056\uff90\057\uff90\060\uff90\061\uff90\062" +
    "\uff90\064\uff90\066\uff90\072\uff90\073\uff90\074\uff90\075\uff90" +
    "\001\002\000\052\043\uff94\044\uff94\045\uff94\046\uff94\047" +
    "\uff94\050\uff94\053\uff94\054\uff94\055\uff94\056\uff94\057\uff94" +
    "\060\uff94\061\uff94\062\uff94\064\uff94\066\uff94\072\uff94\073" +
    "\uff94\074\uff94\075\uff94\001\002\000\042\012\uffad\013\uffad" +
    "\015\uffad\016\uffad\017\uffad\020\uffad\021\uffad\022\uffad\023" +
    "\uffad\024\uffad\025\uffad\027\uffad\030\uffad\036\uffad\070\uffad" +
    "\100\uffad\001\002\000\052\043\uff91\044\uff91\045\uff91\046" +
    "\uff91\047\uff91\050\uff91\053\uff91\054\uff91\055\uff91\056\uff91" +
    "\057\uff91\060\uff91\061\uff91\062\uff91\064\uff91\066\uff91\072" +
    "\uff91\073\uff91\074\uff91\075\uff91\001\002\000\036\031\105" +
    "\032\117\033\102\041\107\042\115\044\106\063\122\065" +
    "\103\071\113\100\116\101\110\102\104\103\120\104\112" +
    "\001\002\000\042\043\134\044\127\045\133\046\137\047" +
    "\130\050\132\053\140\054\141\055\131\056\124\057\136" +
    "\060\135\061\125\062\142\064\126\073\212\001\002\000" +
    "\052\043\uff8f\044\uff8f\045\uff8f\046\uff8f\047\uff8f\050\uff8f" +
    "\053\uff8f\054\uff8f\055\uff8f\056\uff8f\057\uff8f\060\uff8f\061" +
    "\uff8f\062\uff8f\064\uff8f\066\uff8f\072\uff8f\073\uff8f\074\uff8f" +
    "\075\uff8f\001\002\000\060\043\uff95\044\uff95\045\uff95\046" +
    "\uff95\047\uff95\050\uff95\053\uff95\054\uff95\055\uff95\056\uff95" +
    "\057\uff95\060\uff95\061\uff95\062\uff95\064\uff95\065\167\066" +
    "\uff95\071\166\072\uff95\073\uff95\074\uff95\075\uff95\076\165" +
    "\001\002\000\004\065\162\001\002\000\052\043\uff92\044" +
    "\uff92\045\uff92\046\uff92\047\uff92\050\uff92\053\uff92\054\uff92" +
    "\055\uff92\056\uff92\057\uff92\060\uff92\061\uff92\062\uff92\064" +
    "\uff92\066\uff92\072\uff92\073\uff92\074\uff92\075\uff92\001\002" +
    "\000\052\043\uff86\044\uff86\045\uff86\046\uff86\047\uff86\050" +
    "\uff86\053\uff86\054\uff86\055\uff86\056\uff86\057\uff86\060\uff86" +
    "\061\uff86\062\uff86\064\uff86\066\uff86\072\uff86\073\uff86\074" +
    "\uff86\075\uff86\001\002\000\036\031\105\032\117\033\102" +
    "\041\107\042\115\044\106\063\122\065\103\071\113\100" +
    "\116\101\110\102\104\103\120\104\112\001\002\000\052" +
    "\043\134\044\127\045\133\046\137\047\130\050\132\053" +
    "\140\054\141\055\131\056\124\057\136\060\135\061\uff97" +
    "\062\uff97\064\uff97\066\uff97\072\uff97\073\uff97\074\uff97\075" +
    "\uff97\001\002\000\036\031\105\032\117\033\102\041\107" +
    "\042\115\044\106\063\122\065\103\071\113\100\116\101" +
    "\110\102\104\103\120\104\112\001\002\000\036\031\105" +
    "\032\117\033\102\041\107\042\115\044\106\063\122\065" +
    "\103\071\113\100\116\101\110\102\104\103\120\104\112" +
    "\001\002\000\036\031\105\032\117\033\102\041\107\042" +
    "\115\044\106\063\122\065\103\071\113\100\116\101\110" +
    "\102\104\103\120\104\112\001\002\000\036\031\105\032" +
    "\117\033\102\041\107\042\115\044\106\063\122\065\103" +
    "\071\113\100\116\101\110\102\104\103\120\104\112\001" +
    "\002\000\036\031\105\032\117\033\102\041\107\042\115" +
    "\044\106\063\122\065\103\071\113\100\116\101\110\102" +
    "\104\103\120\104\112\001\002\000\036\031\105\032\117" +
    "\033\102\041\107\042\115\044\106\063\122\065\103\071" +
    "\113\100\116\101\110\102\104\103\120\104\112\001\002" +
    "\000\036\031\105\032\117\033\102\041\107\042\115\044" +
    "\106\063\122\065\103\071\113\100\116\101\110\102\104" +
    "\103\120\104\112\001\002\000\036\031\105\032\117\033" +
    "\102\041\107\042\115\044\106\063\122\065\103\071\113" +
    "\100\116\101\110\102\104\103\120\104\112\001\002\000" +
    "\036\031\105\032\117\033\102\041\107\042\115\044\106" +
    "\063\122\065\103\071\113\100\116\101\110\102\104\103" +
    "\120\104\112\001\002\000\036\031\105\032\117\033\102" +
    "\041\107\042\115\044\106\063\122\065\103\071\113\100" +
    "\116\101\110\102\104\103\120\104\112\001\002\000\036" +
    "\031\105\032\117\033\102\041\107\042\115\044\106\063" +
    "\122\065\103\071\113\100\116\101\110\102\104\103\120" +
    "\104\112\001\002\000\036\031\105\032\117\033\102\041" +
    "\107\042\115\044\106\063\122\065\103\071\113\100\116" +
    "\101\110\102\104\103\120\104\112\001\002\000\036\031" +
    "\105\032\117\033\102\041\107\042\115\044\106\063\122" +
    "\065\103\071\113\100\116\101\110\102\104\103\120\104" +
    "\112\001\002\000\036\031\105\032\117\033\102\041\107" +
    "\042\115\044\106\063\122\065\103\071\113\100\116\101" +
    "\110\102\104\103\120\104\112\001\002\000\036\031\105" +
    "\032\117\033\102\041\107\042\115\044\106\063\122\065" +
    "\103\071\113\100\116\101\110\102\104\103\120\104\112" +
    "\001\002\000\052\043\134\044\127\045\133\046\137\047" +
    "\130\050\132\053\140\054\141\055\131\056\124\057\136" +
    "\060\135\061\125\062\uff99\064\126\066\uff99\072\uff99\073" +
    "\uff99\074\uff99\075\uff99\001\002\000\052\043\134\044\127" +
    "\045\133\046\137\047\130\050\132\053\uff9f\054\uff9f\055" +
    "\uff9f\056\uff9f\057\uff9f\060\uff9f\061\uff9f\062\uff9f\064\uff9f" +
    "\066\uff9f\072\uff9f\073\uff9f\074\uff9f\075\uff9f\001\002\000" +
    "\052\043\134\044\127\045\133\046\137\047\130\050\132" +
    "\053\uffa0\054\uffa0\055\uffa0\056\uffa0\057\uffa0\060\uffa0\061" +
    "\uffa0\062\uffa0\064\uffa0\066\uffa0\072\uffa0\073\uffa0\074\uffa0" +
    "\075\uffa0\001\002\000\052\043\uffa3\044\uffa3\045\uffa3\046" +
    "\uffa3\047\130\050\uffa3\053\uffa3\054\uffa3\055\uffa3\056\uffa3" +
    "\057\uffa3\060\uffa3\061\uffa3\062\uffa3\064\uffa3\066\uffa3\072" +
    "\uffa3\073\uffa3\074\uffa3\075\uffa3\001\002\000\052\043\134" +
    "\044\127\045\133\046\137\047\130\050\132\053\uff9d\054" +
    "\uff9d\055\uff9d\056\uff9d\057\uff9d\060\uff9d\061\uff9d\062\uff9d" +
    "\064\uff9d\066\uff9d\072\uff9d\073\uff9d\074\uff9d\075\uff9d\001" +
    "\002\000\052\043\134\044\127\045\133\046\137\047\130" +
    "\050\132\053\uff9b\054\uff9b\055\uff9b\056\uff9b\057\uff9b\060" +
    "\uff9b\061\uff9b\062\uff9b\064\uff9b\066\uff9b\072\uff9b\073\uff9b" +
    "\074\uff9b\075\uff9b\001\002\000\052\043\uffa6\044\uffa6\045" +
    "\133\046\137\047\130\050\132\053\uffa6\054\uffa6\055\uffa6" +
    "\056\uffa6\057\uffa6\060\uffa6\061\uffa6\062\uffa6\064\uffa6\066" +
    "\uffa6\072\uffa6\073\uffa6\074\uffa6\075\uffa6\001\002\000\052" +
    "\043\uffa4\044\uffa4\045\uffa4\046\uffa4\047\130\050\uffa4\053" +
    "\uffa4\054\uffa4\055\uffa4\056\uffa4\057\uffa4\060\uffa4\061\uffa4" +
    "\062\uffa4\064\uffa4\066\uffa4\072\uffa4\073\uffa4\074\uffa4\075" +
    "\uffa4\001\002\000\052\043\uffa1\044\uffa1\045\uffa1\046\uffa1" +
    "\047\130\050\uffa1\053\uffa1\054\uffa1\055\uffa1\056\uffa1\057" +
    "\uffa1\060\uffa1\061\uffa1\062\uffa1\064\uffa1\066\uffa1\072\uffa1" +
    "\073\uffa1\074\uffa1\075\uffa1\001\002\000\052\043\134\044" +
    "\127\045\133\046\137\047\130\050\132\053\uff9e\054\uff9e" +
    "\055\uff9e\056\uff9e\057\uff9e\060\uff9e\061\uff9e\062\uff9e\064" +
    "\uff9e\066\uff9e\072\uff9e\073\uff9e\074\uff9e\075\uff9e\001\002" +
    "\000\052\043\uffa2\044\uffa2\045\uffa2\046\uffa2\047\130\050" +
    "\uffa2\053\uffa2\054\uffa2\055\uffa2\056\uffa2\057\uffa2\060\uffa2" +
    "\061\uffa2\062\uffa2\064\uffa2\066\uffa2\072\uffa2\073\uffa2\074" +
    "\uffa2\075\uffa2\001\002\000\052\043\uffa5\044\uffa5\045\133" +
    "\046\137\047\130\050\132\053\uffa5\054\uffa5\055\uffa5\056" +
    "\uffa5\057\uffa5\060\uffa5\061\uffa5\062\uffa5\064\uffa5\066\uffa5" +
    "\072\uffa5\073\uffa5\074\uffa5\075\uffa5\001\002\000\052\043" +
    "\134\044\127\045\133\046\137\047\130\050\132\053\140" +
    "\054\141\055\131\056\124\057\136\060\135\061\uff98\062" +
    "\uff98\064\uff98\066\uff98\072\uff98\073\uff98\074\uff98\075\uff98" +
    "\001\002\000\052\043\134\044\127\045\133\046\137\047" +
    "\130\050\132\053\140\054\141\055\131\056\124\057\136" +
    "\060\135\061\uff9a\062\uff9a\064\126\066\uff9a\072\uff9a\073" +
    "\uff9a\074\uff9a\075\uff9a\001\002\000\052\043\134\044\127" +
    "\045\133\046\137\047\130\050\132\053\uff9c\054\uff9c\055" +
    "\uff9c\056\uff9c\057\uff9c\060\uff9c\061\uff9c\062\uff9c\064\uff9c" +
    "\066\uff9c\072\uff9c\073\uff9c\074\uff9c\075\uff9c\001\002\000" +
    "\036\031\105\032\117\033\102\041\107\042\115\044\106" +
    "\063\122\065\103\071\113\100\116\101\110\102\104\103" +
    "\120\104\112\001\002\000\042\043\134\044\127\045\133" +
    "\046\137\047\130\050\132\053\140\054\141\055\131\056" +
    "\124\057\136\060\135\061\125\062\142\064\126\066\164" +
    "\001\002\000\052\043\uff8b\044\uff8b\045\uff8b\046\uff8b\047" +
    "\uff8b\050\uff8b\053\uff8b\054\uff8b\055\uff8b\056\uff8b\057\uff8b" +
    "\060\uff8b\061\uff8b\062\uff8b\064\uff8b\066\uff8b\072\uff8b\073" +
    "\uff8b\074\uff8b\075\uff8b\001\002\000\006\034\202\040\203" +
    "\001\002\000\036\031\105\032\117\033\102\041\107\042" +
    "\115\044\106\063\122\065\103\071\113\100\116\101\110" +
    "\102\104\103\120\104\112\001\002\000\042\031\105\032" +
    "\117\033\102\041\107\042\115\044\106\063\122\065\103" +
    "\066\uff82\071\113\075\uff82\100\116\101\110\102\104\103" +
    "\120\104\112\001\002\000\044\043\134\044\127\045\133" +
    "\046\137\047\130\050\132\053\140\054\141\055\131\056" +
    "\124\057\136\060\135\061\125\062\142\064\126\066\uff83" +
    "\075\uff83\001\002\000\006\066\173\075\172\001\002\000" +
    "\036\031\105\032\117\033\102\041\107\042\115\044\106" +
    "\063\122\065\103\071\113\100\116\101\110\102\104\103" +
    "\120\104\112\001\002\000\052\043\uff85\044\uff85\045\uff85" +
    "\046\uff85\047\uff85\050\uff85\053\uff85\054\uff85\055\uff85\056" +
    "\uff85\057\uff85\060\uff85\061\uff85\062\uff85\064\uff85\066\uff85" +
    "\072\uff85\073\uff85\074\uff85\075\uff85\001\002\000\044\043" +
    "\134\044\127\045\133\046\137\047\130\050\132\053\140" +
    "\054\141\055\131\056\124\057\136\060\135\061\125\062" +
    "\142\064\126\066\uff84\075\uff84\001\002\000\042\043\134" +
    "\044\127\045\133\046\137\047\130\050\132\053\140\054" +
    "\141\055\131\056\124\057\136\060\135\061\125\062\142" +
    "\064\126\072\176\001\002\000\054\043\uff8d\044\uff8d\045" +
    "\uff8d\046\uff8d\047\uff8d\050\uff8d\053\uff8d\054\uff8d\055\uff8d" +
    "\056\uff8d\057\uff8d\060\uff8d\061\uff8d\062\uff8d\064\uff8d\066" +
    "\uff8d\071\177\072\uff8d\073\uff8d\074\uff8d\075\uff8d\001\002" +
    "\000\036\031\105\032\117\033\102\041\107\042\115\044" +
    "\106\063\122\065\103\071\113\100\116\101\110\102\104" +
    "\103\120\104\112\001\002\000\042\043\134\044\127\045" +
    "\133\046\137\047\130\050\132\053\140\054\141\055\131" +
    "\056\124\057\136\060\135\061\125\062\142\064\126\072" +
    "\201\001\002\000\052\043\uff8c\044\uff8c\045\uff8c\046\uff8c" +
    "\047\uff8c\050\uff8c\053\uff8c\054\uff8c\055\uff8c\056\uff8c\057" +
    "\uff8c\060\uff8c\061\uff8c\062\uff8c\064\uff8c\066\uff8c\072\uff8c" +
    "\073\uff8c\074\uff8c\075\uff8c\001\002\000\004\065\207\001" +
    "\002\000\004\065\204\001\002\000\036\031\105\032\117" +
    "\033\102\041\107\042\115\044\106\063\122\065\103\071" +
    "\113\100\116\101\110\102\104\103\120\104\112\001\002" +
    "\000\042\043\134\044\127\045\133\046\137\047\130\050" +
    "\132\053\140\054\141\055\131\056\124\057\136\060\135" +
    "\061\125\062\142\064\126\066\206\001\002\000\052\043" +
    "\uff87\044\uff87\045\uff87\046\uff87\047\uff87\050\uff87\053\uff87" +
    "\054\uff87\055\uff87\056\uff87\057\uff87\060\uff87\061\uff87\062" +
    "\uff87\064\uff87\066\uff87\072\uff87\073\uff87\074\uff87\075\uff87" +
    "\001\002\000\036\031\105\032\117\033\102\041\107\042" +
    "\115\044\106\063\122\065\103\071\113\100\116\101\110" +
    "\102\104\103\120\104\112\001\002\000\042\043\134\044" +
    "\127\045\133\046\137\047\130\050\132\053\140\054\141" +
    "\055\131\056\124\057\136\060\135\061\125\062\142\064" +
    "\126\066\211\001\002\000\052\043\uff88\044\uff88\045\uff88" +
    "\046\uff88\047\uff88\050\uff88\053\uff88\054\uff88\055\uff88\056" +
    "\uff88\057\uff88\060\uff88\061\uff88\062\uff88\064\uff88\066\uff88" +
    "\072\uff88\073\uff88\074\uff88\075\uff88\001\002\000\042\012" +
    "\uffae\013\uffae\015\uffae\016\uffae\017\uffae\020\uffae\021\uffae" +
    "\022\uffae\023\uffae\024\uffae\025\uffae\027\uffae\030\uffae\036" +
    "\uffae\070\uffae\100\uffae\001\002\000\006\072\216\075\215" +
    "\001\002\000\044\043\134\044\127\045\133\046\137\047" +
    "\130\050\132\053\140\054\141\055\131\056\124\057\136" +
    "\060\135\061\125\062\142\064\126\072\uffc6\075\uffc6\001" +
    "\002\000\036\031\105\032\117\033\102\041\107\042\115" +
    "\044\106\063\122\065\103\071\113\100\116\101\110\102" +
    "\104\103\120\104\112\001\002\000\052\043\uff8e\044\uff8e" +
    "\045\uff8e\046\uff8e\047\uff8e\050\uff8e\053\uff8e\054\uff8e\055" +
    "\uff8e\056\uff8e\057\uff8e\060\uff8e\061\uff8e\062\uff8e\064\uff8e" +
    "\066\uff8e\072\uff8e\073\uff8e\074\uff8e\075\uff8e\001\002\000" +
    "\044\043\134\044\127\045\133\046\137\047\130\050\132" +
    "\053\140\054\141\055\131\056\124\057\136\060\135\061" +
    "\125\062\142\064\126\072\uffc7\075\uffc7\001\002\000\052" +
    "\043\uffac\044\uffac\045\uffac\046\uffac\047\130\050\uffac\053" +
    "\uffac\054\uffac\055\uffac\056\uffac\057\uffac\060\uffac\061\uffac" +
    "\062\uffac\064\uffac\066\uffac\072\uffac\073\uffac\074\uffac\075" +
    "\uffac\001\002\000\036\031\105\032\117\033\102\041\107" +
    "\042\115\044\106\063\122\065\103\071\113\100\116\101" +
    "\110\102\104\103\120\104\112\001\002\000\042\043\134" +
    "\044\127\045\133\046\137\047\130\050\132\053\140\054" +
    "\141\055\131\056\124\057\136\060\135\061\125\062\142" +
    "\064\126\066\223\001\002\000\052\043\uff8a\044\uff8a\045" +
    "\uff8a\046\uff8a\047\uff8a\050\uff8a\053\uff8a\054\uff8a\055\uff8a" +
    "\056\uff8a\057\uff8a\060\uff8a\061\uff8a\062\uff8a\064\uff8a\066" +
    "\uff8a\072\uff8a\073\uff8a\074\uff8a\075\uff8a\001\002\000\004" +
    "\066\243\001\002\000\004\066\241\001\002\000\042\043" +
    "\134\044\127\045\133\046\137\047\130\050\132\053\140" +
    "\054\141\055\131\056\124\057\136\060\135\061\125\062" +
    "\142\064\126\066\240\001\002\000\004\066\236\001\002" +
    "\000\004\066\234\001\002\000\004\066\232\001\002\000" +
    "\036\031\105\032\117\033\102\041\107\042\115\044\106" +
    "\063\122\065\103\071\113\100\116\101\110\102\104\103" +
    "\120\104\112\001\002\000\052\043\uffaa\044\uffaa\045\uffaa" +
    "\046\uffaa\047\130\050\uffaa\053\uffaa\054\uffaa\055\uffaa\056" +
    "\uffaa\057\uffaa\060\uffaa\061\uffaa\062\uffaa\064\uffaa\066\uffaa" +
    "\072\uffaa\073\uffaa\074\uffaa\075\uffaa\001\002\000\036\031" +
    "\105\032\117\033\102\041\107\042\115\044\106\063\122" +
    "\065\103\071\113\100\116\101\110\102\104\103\120\104" +
    "\112\001\002\000\052\043\uffa7\044\uffa7\045\uffa7\046\uffa7" +
    "\047\130\050\uffa7\053\uffa7\054\uffa7\055\uffa7\056\uffa7\057" +
    "\uffa7\060\uffa7\061\uffa7\062\uffa7\064\uffa7\066\uffa7\072\uffa7" +
    "\073\uffa7\074\uffa7\075\uffa7\001\002\000\036\031\105\032" +
    "\117\033\102\041\107\042\115\044\106\063\122\065\103" +
    "\071\113\100\116\101\110\102\104\103\120\104\112\001" +
    "\002\000\052\043\uffa8\044\uffa8\045\uffa8\046\uffa8\047\130" +
    "\050\uffa8\053\uffa8\054\uffa8\055\uffa8\056\uffa8\057\uffa8\060" +
    "\uffa8\061\uffa8\062\uffa8\064\uffa8\066\uffa8\072\uffa8\073\uffa8" +
    "\074\uffa8\075\uffa8\001\002\000\052\043\uff96\044\uff96\045" +
    "\uff96\046\uff96\047\uff96\050\uff96\053\uff96\054\uff96\055\uff96" +
    "\056\uff96\057\uff96\060\uff96\061\uff96\062\uff96\064\uff96\066" +
    "\uff96\072\uff96\073\uff96\074\uff96\075\uff96\001\002\000\036" +
    "\031\105\032\117\033\102\041\107\042\115\044\106\063" +
    "\122\065\103\071\113\100\116\101\110\102\104\103\120" +
    "\104\112\001\002\000\052\043\uffa9\044\uffa9\045\uffa9\046" +
    "\uffa9\047\130\050\uffa9\053\uffa9\054\uffa9\055\uffa9\056\uffa9" +
    "\057\uffa9\060\uffa9\061\uffa9\062\uffa9\064\uffa9\066\uffa9\072" +
    "\uffa9\073\uffa9\074\uffa9\075\uffa9\001\002\000\036\031\105" +
    "\032\117\033\102\041\107\042\115\044\106\063\122\065" +
    "\103\071\113\100\116\101\110\102\104\103\120\104\112" +
    "\001\002\000\052\043\uffab\044\uffab\045\uffab\046\uffab\047" +
    "\130\050\uffab\053\uffab\054\uffab\055\uffab\056\uffab\057\uffab" +
    "\060\uffab\061\uffab\062\uffab\064\uffab\066\uffab\072\uffab\073" +
    "\uffab\074\uffab\075\uffab\001\002\000\036\031\105\032\117" +
    "\033\102\041\107\042\115\044\106\063\122\065\103\071" +
    "\113\100\116\101\110\102\104\103\120\104\112\001\002" +
    "\000\042\043\134\044\127\045\133\046\137\047\130\050" +
    "\132\053\140\054\141\055\131\056\124\057\136\060\135" +
    "\061\125\062\142\064\126\066\247\001\002\000\052\043" +
    "\uff89\044\uff89\045\uff89\046\uff89\047\uff89\050\uff89\053\uff89" +
    "\054\uff89\055\uff89\056\uff89\057\uff89\060\uff89\061\uff89\062" +
    "\uff89\064\uff89\066\uff89\072\uff89\073\uff89\074\uff89\075\uff89" +
    "\001\002\000\042\012\uffe5\013\uffe5\015\uffe5\016\uffe5\017" +
    "\uffe5\020\uffe5\021\uffe5\022\uffe5\023\uffe5\024\uffe5\025\uffe5" +
    "\027\uffe5\030\uffe5\036\uffe5\070\uffe5\100\uffe5\001\002\000" +
    "\004\020\252\001\002\000\004\065\253\001\002\000\036" +
    "\031\105\032\117\033\102\041\107\042\115\044\106\063" +
    "\122\065\103\071\113\100\116\101\110\102\104\103\120" +
    "\104\112\001\002\000\042\043\134\044\127\045\133\046" +
    "\137\047\130\050\132\053\140\054\141\055\131\056\124" +
    "\057\136\060\135\061\125\062\142\064\126\066\255\001" +
    "\002\000\004\073\256\001\002\000\062\002\uffb5\004\uffb5" +
    "\005\uffb5\006\uffb5\007\uffb5\010\uffb5\011\uffb5\012\uffb5\013" +
    "\uffb5\015\uffb5\016\uffb5\017\uffb5\020\uffb5\021\uffb5\022\uffb5" +
    "\023\uffb5\024\uffb5\025\uffb5\026\uffb5\027\uffb5\030\uffb5\036" +
    "\uffb5\070\uffb5\100\uffb5\001\002\000\036\031\105\032\117" +
    "\033\102\041\107\042\115\044\106\063\122\065\103\071" +
    "\113\100\116\101\110\102\104\103\120\104\112\001\002" +
    "\000\042\043\134\044\127\045\133\046\137\047\130\050" +
    "\132\053\140\054\141\055\131\056\124\057\136\060\135" +
    "\061\125\062\142\064\126\066\261\001\002\000\004\073" +
    "\262\001\002\000\062\002\uffc0\004\uffc0\005\uffc0\006\uffc0" +
    "\007\uffc0\010\uffc0\011\uffc0\012\uffc0\013\uffc0\015\uffc0\016" +
    "\uffc0\017\uffc0\020\uffc0\021\uffc0\022\uffc0\023\uffc0\024\uffc0" +
    "\025\uffc0\026\uffc0\027\uffc0\030\uffc0\036\uffc0\070\uffc0\100" +
    "\uffc0\001\002\000\004\074\264\001\002\000\014\004\012" +
    "\005\010\006\016\007\006\010\045\001\002\000\010\071" +
    "\266\073\270\077\267\001\002\000\004\072\273\001\002" +
    "\000\036\031\105\032\117\033\102\041\107\042\115\044" +
    "\106\063\122\065\103\071\113\100\116\101\110\102\104" +
    "\103\120\104\112\001\002\000\062\002\uffd2\004\uffd2\005" +
    "\uffd2\006\uffd2\007\uffd2\010\uffd2\011\uffd2\012\uffd2\013\uffd2" +
    "\015\uffd2\016\uffd2\017\uffd2\020\uffd2\021\uffd2\022\uffd2\023" +
    "\uffd2\024\uffd2\025\uffd2\026\uffd2\027\uffd2\030\uffd2\036\uffd2" +
    "\070\uffd2\100\uffd2\001\002\000\042\043\134\044\127\045" +
    "\133\046\137\047\130\050\132\053\140\054\141\055\131" +
    "\056\124\057\136\060\135\061\125\062\142\064\126\073" +
    "\272\001\002\000\062\002\uffd3\004\uffd3\005\uffd3\006\uffd3" +
    "\007\uffd3\010\uffd3\011\uffd3\012\uffd3\013\uffd3\015\uffd3\016" +
    "\uffd3\017\uffd3\020\uffd3\021\uffd3\022\uffd3\023\uffd3\024\uffd3" +
    "\025\uffd3\026\uffd3\027\uffd3\030\uffd3\036\uffd3\070\uffd3\100" +
    "\uffd3\001\002\000\006\071\274\077\275\001\002\000\004" +
    "\072\302\001\002\000\004\071\276\001\002\000\036\031" +
    "\105\032\117\033\102\041\107\042\115\044\106\063\122" +
    "\065\103\071\113\100\116\101\110\102\104\103\120\104" +
    "\112\001\002\000\006\072\300\075\215\001\002\000\004" +
    "\073\301\001\002\000\062\002\uffc9\004\uffc9\005\uffc9\006" +
    "\uffc9\007\uffc9\010\uffc9\011\uffc9\012\uffc9\013\uffc9\015\uffc9" +
    "\016\uffc9\017\uffc9\020\uffc9\021\uffc9\022\uffc9\023\uffc9\024" +
    "\uffc9\025\uffc9\026\uffc9\027\uffc9\030\uffc9\036\uffc9\070\uffc9" +
    "\100\uffc9\001\002\000\004\077\303\001\002\000\004\071" +
    "\304\001\002\000\036\031\105\032\117\033\102\041\107" +
    "\042\115\044\106\063\122\065\103\071\113\100\116\101" +
    "\110\102\104\103\120\104\112\001\002\000\006\072\306" +
    "\075\215\001\002\000\004\073\307\001\002\000\062\002" +
    "\uffc8\004\uffc8\005\uffc8\006\uffc8\007\uffc8\010\uffc8\011\uffc8" +
    "\012\uffc8\013\uffc8\015\uffc8\016\uffc8\017\uffc8\020\uffc8\021" +
    "\uffc8\022\uffc8\023\uffc8\024\uffc8\025\uffc8\026\uffc8\027\uffc8" +
    "\030\uffc8\036\uffc8\070\uffc8\100\uffc8\001\002\000\004\002" +
    "\001\001\002\000\014\004\012\005\010\006\016\007\006" +
    "\010\045\001\002\000\004\057\313\001\002\000\004\100" +
    "\314\001\002\000\004\077\315\001\002\000\004\035\316" +
    "\001\002\000\004\036\317\001\002\000\004\065\320\001" +
    "\002\000\004\066\321\001\002\000\004\073\322\001\002" +
    "\000\062\002\uffc3\004\uffc3\005\uffc3\006\uffc3\007\uffc3\010" +
    "\uffc3\011\uffc3\012\uffc3\013\uffc3\015\uffc3\016\uffc3\017\uffc3" +
    "\020\uffc3\021\uffc3\022\uffc3\023\uffc3\024\uffc3\025\uffc3\026" +
    "\uffc3\027\uffc3\030\uffc3\036\uffc3\070\uffc3\100\uffc3\001\002" +
    "\000\036\031\105\032\117\033\102\041\107\042\115\044" +
    "\106\063\122\065\103\071\113\100\116\101\110\102\104" +
    "\103\120\104\112\001\002\000\042\043\134\044\127\045" +
    "\133\046\137\047\130\050\132\053\140\054\141\055\131" +
    "\056\124\057\136\060\135\061\125\062\142\064\126\066" +
    "\325\001\002\000\004\067\326\001\002\000\034\012\051" +
    "\013\022\015\032\020\040\021\053\022\014\023\026\024" +
    "\011\025\061\027\052\030\013\036\041\100\025\001\002" +
    "\000\036\012\051\013\022\015\032\020\040\021\053\022" +
    "\014\023\026\024\011\025\061\027\052\030\013\036\041" +
    "\070\330\100\025\001\002\000\062\002\uffb6\004\uffb6\005" +
    "\uffb6\006\uffb6\007\uffb6\010\uffb6\011\uffb6\012\uffb6\013\uffb6" +
    "\015\uffb6\016\uffb6\017\uffb6\020\uffb6\021\uffb6\022\uffb6\023" +
    "\uffb6\024\uffb6\025\uffb6\026\uffb6\027\uffb6\030\uffb6\036\uffb6" +
    "\070\uffb6\100\uffb6\001\002\000\004\065\332\001\002\000" +
    "\004\066\333\001\002\000\004\073\334\001\002\000\052" +
    "\002\uffec\004\uffec\005\uffec\006\uffec\007\uffec\010\uffec\011" +
    "\uffec\012\uffec\013\uffec\015\uffec\020\uffec\021\uffec\022\uffec" +
    "\023\uffec\024\uffec\026\uffec\027\uffec\030\uffec\036\uffec\100" +
    "\uffec\001\002\000\004\065\336\001\002\000\020\004\012" +
    "\005\010\006\016\007\006\010\045\066\uffe7\075\uffe7\001" +
    "\002\000\004\100\350\001\002\000\006\066\343\075\342" +
    "\001\002\000\006\066\uffe8\075\uffe8\001\002\000\014\004" +
    "\012\005\010\006\016\007\006\010\045\001\002\000\004" +
    "\067\344\001\002\000\034\012\051\013\022\015\032\020" +
    "\040\021\053\022\014\023\026\024\011\025\061\027\052" +
    "\030\013\036\041\100\025\001\002\000\036\012\051\013" +
    "\022\015\032\020\040\021\053\022\014\023\026\024\011" +
    "\025\061\027\052\030\013\036\041\070\346\100\025\001" +
    "\002\000\052\002\uffeb\004\uffeb\005\uffeb\006\uffeb\007\uffeb" +
    "\010\uffeb\011\uffeb\012\uffeb\013\uffeb\015\uffeb\020\uffeb\021" +
    "\uffeb\022\uffeb\023\uffeb\024\uffeb\026\uffeb\027\uffeb\030\uffeb" +
    "\036\uffeb\100\uffeb\001\002\000\006\066\uffe9\075\uffe9\001" +
    "\002\000\006\066\uffe6\075\uffe6\001\002\000\052\002\uffed" +
    "\004\uffed\005\uffed\006\uffed\007\uffed\010\uffed\011\uffed\012" +
    "\uffed\013\uffed\015\uffed\020\uffed\021\uffed\022\uffed\023\uffed" +
    "\024\uffed\026\uffed\027\uffed\030\uffed\036\uffed\100\uffed\001" +
    "\002\000\036\031\105\032\117\033\102\041\107\042\115" +
    "\044\106\063\122\065\103\071\113\100\116\101\110\102" +
    "\104\103\120\104\112\001\002\000\042\043\134\044\127" +
    "\045\133\046\137\047\130\050\132\053\140\054\141\055" +
    "\131\056\124\057\136\060\135\061\125\062\142\064\126" +
    "\066\354\001\002\000\004\067\355\001\002\000\006\016" +
    "\361\017\356\001\002\000\004\074\367\001\002\000\010" +
    "\016\361\017\356\070\366\001\002\000\010\016\uffb9\017" +
    "\uffb9\070\uffb9\001\002\000\036\031\105\032\117\033\102" +
    "\041\107\042\115\044\106\063\122\065\103\071\113\100" +
    "\116\101\110\102\104\103\120\104\112\001\002\000\042" +
    "\043\134\044\127\045\133\046\137\047\130\050\132\053" +
    "\140\054\141\055\131\056\124\057\136\060\135\061\125" +
    "\062\142\064\126\074\363\001\002\000\034\012\051\013" +
    "\022\015\032\020\040\021\053\022\014\023\026\024\011" +
    "\025\061\027\052\030\013\036\041\100\025\001\002\000" +
    "\042\012\051\013\022\015\032\016\uffb8\017\uffb8\020\040" +
    "\021\053\022\014\023\026\024\011\025\061\027\052\030" +
    "\013\036\041\070\uffb8\100\025\001\002\000\010\016\uffba" +
    "\017\uffba\070\uffba\001\002\000\062\002\uffbb\004\uffbb\005" +
    "\uffbb\006\uffbb\007\uffbb\010\uffbb\011\uffbb\012\uffbb\013\uffbb" +
    "\015\uffbb\016\uffbb\017\uffbb\020\uffbb\021\uffbb\022\uffbb\023" +
    "\uffbb\024\uffbb\025\uffbb\026\uffbb\027\uffbb\030\uffbb\036\uffbb" +
    "\070\uffbb\100\uffbb\001\002\000\034\012\051\013\022\015" +
    "\032\020\040\021\053\022\014\023\026\024\011\025\061" +
    "\027\052\030\013\036\041\100\025\001\002\000\042\012" +
    "\051\013\022\015\032\016\uffb7\017\uffb7\020\040\021\053" +
    "\022\014\023\026\024\011\025\061\027\052\030\013\036" +
    "\041\070\uffb7\100\025\001\002\000\004\065\372\001\002" +
    "\000\020\004\012\005\010\006\016\007\006\010\045\066" +
    "\uffe7\075\uffe7\001\002\000\006\066\374\075\342\001\002" +
    "\000\004\067\375\001\002\000\034\012\051\013\022\015" +
    "\032\020\040\021\053\022\014\023\026\024\011\025\061" +
    "\027\052\030\013\036\041\100\025\001\002\000\036\012" +
    "\051\013\022\015\032\020\040\021\053\022\014\023\026" +
    "\024\011\025\061\027\052\030\013\036\041\070\377\100" +
    "\025\001\002\000\052\002\uffea\004\uffea\005\uffea\006\uffea" +
    "\007\uffea\010\uffea\011\uffea\012\uffea\013\uffea\015\uffea\020" +
    "\uffea\021\uffea\022\uffea\023\uffea\024\uffea\026\uffea\027\uffea" +
    "\030\uffea\036\uffea\100\uffea\001\002\000\062\002\uffb0\004" +
    "\uffb0\005\uffb0\006\uffb0\007\uffb0\010\uffb0\011\uffb0\012\uffb0" +
    "\013\uffb0\015\uffb0\016\uffb0\017\uffb0\020\uffb0\021\uffb0\022" +
    "\uffb0\023\uffb0\024\uffb0\025\uffb0\026\uffb0\027\uffb0\030\uffb0" +
    "\036\uffb0\070\uffb0\100\uffb0\001\002\000\004\073\u011c\001" +
    "\002\000\006\037\u0114\040\u0115\001\002\000\004\073\u0113" +
    "\001\002\000\036\031\105\032\117\033\102\041\107\042" +
    "\115\044\106\063\122\065\103\071\113\100\116\101\110" +
    "\102\104\103\120\104\112\001\002\000\036\031\105\032" +
    "\117\033\102\041\107\042\115\044\106\063\122\065\103" +
    "\071\113\100\116\101\110\102\104\103\120\104\112\001" +
    "\002\000\042\043\134\044\127\045\133\046\137\047\130" +
    "\050\132\053\140\054\141\055\131\056\124\057\136\060" +
    "\135\061\125\062\142\064\126\073\u0107\001\002\000\114" +
    "\002\uffcc\004\uffcc\005\uffcc\006\uffcc\007\uffcc\010\uffcc\011" +
    "\uffcc\012\uffcc\013\uffcc\015\uffcc\016\uffcc\017\uffcc\020\uffcc" +
    "\021\uffcc\022\uffcc\023\uffcc\024\uffcc\025\uffcc\026\uffcc\027" +
    "\uffcc\030\uffcc\031\uffcc\032\uffcc\033\uffcc\036\uffcc\041\uffcc" +
    "\042\uffcc\044\uffcc\063\uffcc\065\uffcc\070\uffcc\071\uffcc\100" +
    "\uffcc\101\uffcc\102\uffcc\103\uffcc\104\uffcc\001\002\000\042" +
    "\043\134\044\127\045\133\046\137\047\130\050\132\053" +
    "\140\054\141\055\131\056\124\057\136\060\135\061\125" +
    "\062\142\064\126\072\u0109\001\002\000\006\071\u010a\077" +
    "\u010b\001\002\000\036\031\105\032\117\033\102\041\107" +
    "\042\115\044\106\063\122\065\103\071\113\100\116\101" +
    "\110\102\104\103\120\104\112\001\002\000\036\031\105" +
    "\032\117\033\102\041\107\042\115\044\106\063\122\065" +
    "\103\071\113\100\116\101\110\102\104\103\120\104\112" +
    "\001\002\000\042\043\134\044\127\045\133\046\137\047" +
    "\130\050\132\053\140\054\141\055\131\056\124\057\136" +
    "\060\135\061\125\062\142\064\126\073\u010d\001\002\000" +
    "\062\002\uffc5\004\uffc5\005\uffc5\006\uffc5\007\uffc5\010\uffc5" +
    "\011\uffc5\012\uffc5\013\uffc5\015\uffc5\016\uffc5\017\uffc5\020" +
    "\uffc5\021\uffc5\022\uffc5\023\uffc5\024\uffc5\025\uffc5\026\uffc5" +
    "\027\uffc5\030\uffc5\036\uffc5\070\uffc5\100\uffc5\001\002\000" +
    "\042\043\134\044\127\045\133\046\137\047\130\050\132" +
    "\053\140\054\141\055\131\056\124\057\136\060\135\061" +
    "\125\062\142\064\126\072\u010f\001\002\000\004\077\u0110" +
    "\001\002\000\036\031\105\032\117\033\102\041\107\042" +
    "\115\044\106\063\122\065\103\071\113\100\116\101\110" +
    "\102\104\103\120\104\112\001\002\000\042\043\134\044" +
    "\127\045\133\046\137\047\130\050\132\053\140\054\141" +
    "\055\131\056\124\057\136\060\135\061\125\062\142\064" +
    "\126\073\u0112\001\002\000\062\002\uffc4\004\uffc4\005\uffc4" +
    "\006\uffc4\007\uffc4\010\uffc4\011\uffc4\012\uffc4\013\uffc4\015" +
    "\uffc4\016\uffc4\017\uffc4\020\uffc4\021\uffc4\022\uffc4\023\uffc4" +
    "\024\uffc4\025\uffc4\026\uffc4\027\uffc4\030\uffc4\036\uffc4\070" +
    "\uffc4\100\uffc4\001\002\000\114\002\uffcb\004\uffcb\005\uffcb" +
    "\006\uffcb\007\uffcb\010\uffcb\011\uffcb\012\uffcb\013\uffcb\015" +
    "\uffcb\016\uffcb\017\uffcb\020\uffcb\021\uffcb\022\uffcb\023\uffcb" +
    "\024\uffcb\025\uffcb\026\uffcb\027\uffcb\030\uffcb\031\uffcb\032" +
    "\uffcb\033\uffcb\036\uffcb\041\uffcb\042\uffcb\044\uffcb\063\uffcb" +
    "\065\uffcb\070\uffcb\071\uffcb\100\uffcb\101\uffcb\102\uffcb\103" +
    "\uffcb\104\uffcb\001\002\000\004\065\u0119\001\002\000\004" +
    "\065\u0116\001\002\000\036\031\105\032\117\033\102\041" +
    "\107\042\115\044\106\063\122\065\103\071\113\100\116" +
    "\101\110\102\104\103\120\104\112\001\002\000\042\043" +
    "\134\044\127\045\133\046\137\047\130\050\132\053\140" +
    "\054\141\055\131\056\124\057\136\060\135\061\125\062" +
    "\142\064\126\066\u0118\001\002\000\004\073\uffc1\001\002" +
    "\000\036\031\105\032\117\033\102\041\107\042\115\044" +
    "\106\063\122\065\103\071\113\100\116\101\110\102\104" +
    "\103\120\104\112\001\002\000\042\043\134\044\127\045" +
    "\133\046\137\047\130\050\132\053\140\054\141\055\131" +
    "\056\124\057\136\060\135\061\125\062\142\064\126\066" +
    "\u011b\001\002\000\004\073\uffc2\001\002\000\114\002\uffca" +
    "\004\uffca\005\uffca\006\uffca\007\uffca\010\uffca\011\uffca\012" +
    "\uffca\013\uffca\015\uffca\016\uffca\017\uffca\020\uffca\021\uffca" +
    "\022\uffca\023\uffca\024\uffca\025\uffca\026\uffca\027\uffca\030" +
    "\uffca\031\uffca\032\uffca\033\uffca\036\uffca\041\uffca\042\uffca" +
    "\044\uffca\063\uffca\065\uffca\070\uffca\071\uffca\100\uffca\101" +
    "\uffca\102\uffca\103\uffca\104\uffca\001\002\000\036\031\105" +
    "\032\117\033\102\041\107\042\115\044\106\063\122\065" +
    "\103\071\113\100\116\101\110\102\104\103\120\104\112" +
    "\001\002\000\042\043\134\044\127\045\133\046\137\047" +
    "\130\050\132\053\140\054\141\055\131\056\124\057\136" +
    "\060\135\061\125\062\142\064\126\066\u011f\001\002\000" +
    "\004\067\u0120\001\002\000\034\012\051\013\022\015\032" +
    "\020\040\021\053\022\014\023\026\024\011\025\061\027" +
    "\052\030\013\036\041\100\025\001\002\000\036\012\051" +
    "\013\022\015\032\020\040\021\053\022\014\023\026\024" +
    "\011\025\061\027\052\030\013\036\041\070\u0122\100\025" +
    "\001\002\000\064\002\uffbe\004\uffbe\005\uffbe\006\uffbe\007" +
    "\uffbe\010\uffbe\011\uffbe\012\uffbe\013\uffbe\014\u0123\015\uffbe" +
    "\016\uffbe\017\uffbe\020\uffbe\021\uffbe\022\uffbe\023\uffbe\024" +
    "\uffbe\025\uffbe\026\uffbe\027\uffbe\030\uffbe\036\uffbe\070\uffbe" +
    "\100\uffbe\001\002\000\006\013\022\067\u0125\001\002\000" +
    "\062\002\uffbc\004\uffbc\005\uffbc\006\uffbc\007\uffbc\010\uffbc" +
    "\011\uffbc\012\uffbc\013\uffbc\015\uffbc\016\uffbc\017\uffbc\020" +
    "\uffbc\021\uffbc\022\uffbc\023\uffbc\024\uffbc\025\uffbc\026\uffbc" +
    "\027\uffbc\030\uffbc\036\uffbc\070\uffbc\100\uffbc\001\002\000" +
    "\034\012\051\013\022\015\032\020\040\021\053\022\014" +
    "\023\026\024\011\025\061\027\052\030\013\036\041\100" +
    "\025\001\002\000\036\012\051\013\022\015\032\020\040" +
    "\021\053\022\014\023\026\024\011\025\061\027\052\030" +
    "\013\036\041\070\u0127\100\025\001\002\000\062\002\uffbd" +
    "\004\uffbd\005\uffbd\006\uffbd\007\uffbd\010\uffbd\011\uffbd\012" +
    "\uffbd\013\uffbd\015\uffbd\016\uffbd\017\uffbd\020\uffbd\021\uffbd" +
    "\022\uffbd\023\uffbd\024\uffbd\025\uffbd\026\uffbd\027\uffbd\030" +
    "\uffbd\036\uffbd\070\uffbd\100\uffbd\001\002\000\052\002\uffff" +
    "\004\uffff\005\uffff\006\uffff\007\uffff\010\uffff\011\uffff\012" +
    "\uffff\013\uffff\015\uffff\020\uffff\021\uffff\022\uffff\023\uffff" +
    "\024\uffff\026\uffff\027\uffff\030\uffff\036\uffff\100\uffff\001" +
    "\002\000\004\100\u012a\001\002\000\010\051\u0103\052\u0101" +
    "\077\u0105\001\002\000\036\031\105\032\117\033\102\041" +
    "\107\042\115\044\106\063\122\065\103\071\113\100\116" +
    "\101\110\102\104\103\120\104\112\001\002\000\042\043" +
    "\134\044\127\045\133\046\137\047\130\050\132\053\140" +
    "\054\141\055\131\056\124\057\136\060\135\061\125\062" +
    "\142\064\126\073\u012d\001\002\000\004\100\u012e\001\002" +
    "\000\010\051\u0135\052\u0134\077\u0136\001\002\000\004\066" +
    "\u0130\001\002\000\004\067\u0131\001\002\000\034\012\051" +
    "\013\022\015\032\020\040\021\053\022\014\023\026\024" +
    "\011\025\061\027\052\030\013\036\041\100\025\001\002" +
    "\000\036\012\051\013\022\015\032\020\040\021\053\022" +
    "\014\023\026\024\011\025\061\027\052\030\013\036\041" +
    "\070\u0133\100\025\001\002\000\062\002\uffb1\004\uffb1\005" +
    "\uffb1\006\uffb1\007\uffb1\010\uffb1\011\uffb1\012\uffb1\013\uffb1" +
    "\015\uffb1\016\uffb1\017\uffb1\020\uffb1\021\uffb1\022\uffb1\023" +
    "\uffb1\024\uffb1\025\uffb1\026\uffb1\027\uffb1\030\uffb1\036\uffb1" +
    "\070\uffb1\100\uffb1\001\002\000\004\066\uffb2\001\002\000" +
    "\004\066\uffb3\001\002\000\036\031\105\032\117\033\102" +
    "\041\107\042\115\044\106\063\122\065\103\071\113\100" +
    "\116\101\110\102\104\103\120\104\112\001\002\000\042" +
    "\043\134\044\127\045\133\046\137\047\130\050\132\053" +
    "\140\054\141\055\131\056\124\057\136\060\135\061\125" +
    "\062\142\064\126\066\uffb4\001\002\000\036\031\105\032" +
    "\117\033\102\041\107\042\115\044\106\063\122\065\103" +
    "\071\113\100\116\101\110\102\104\103\120\104\112\001" +
    "\002\000\042\043\134\044\127\045\133\046\137\047\130" +
    "\050\132\053\140\054\141\055\131\056\124\057\136\060" +
    "\135\061\125\062\142\064\126\066\u013a\001\002\000\004" +
    "\073\u013b\001\002\000\062\002\uffbf\004\uffbf\005\uffbf\006" +
    "\uffbf\007\uffbf\010\uffbf\011\uffbf\012\uffbf\013\uffbf\015\uffbf" +
    "\016\uffbf\017\uffbf\020\uffbf\021\uffbf\022\uffbf\023\uffbf\024" +
    "\uffbf\025\uffbf\026\uffbf\027\uffbf\030\uffbf\036\uffbf\070\uffbf" +
    "\100\uffbf\001\002\000\062\002\uffaf\004\uffaf\005\uffaf\006" +
    "\uffaf\007\uffaf\010\uffaf\011\uffaf\012\uffaf\013\uffaf\015\uffaf" +
    "\016\uffaf\017\uffaf\020\uffaf\021\uffaf\022\uffaf\023\uffaf\024" +
    "\uffaf\025\uffaf\026\uffaf\027\uffaf\030\uffaf\036\uffaf\070\uffaf" +
    "\100\uffaf\001\002\000\052\002\uffee\004\uffee\005\uffee\006" +
    "\uffee\007\uffee\010\uffee\011\uffee\012\uffee\013\uffee\015\uffee" +
    "\020\uffee\021\uffee\022\uffee\023\uffee\024\uffee\026\uffee\027" +
    "\uffee\030\uffee\036\uffee\100\uffee\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\u013b\000\054\002\042\003\017\004\043\007\046\010" +
    "\047\011\034\012\030\014\004\016\041\021\020\022\014" +
    "\023\022\025\045\026\003\027\023\030\033\031\016\034" +
    "\035\035\006\036\026\037\032\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\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\050\004\u0127\007\046\010\047" +
    "\011\034\012\030\014\004\016\041\021\020\022\014\023" +
    "\022\025\045\026\003\027\023\030\033\031\016\034\035" +
    "\035\006\036\026\037\032\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\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\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" +
    "\046\005\054\006\061\007\065\010\063\014\067\015\076" +
    "\021\071\022\057\023\055\025\074\026\075\027\066\030" +
    "\056\031\062\034\064\035\073\036\072\037\070\001\001" +
    "\000\044\006\247\007\065\010\063\014\067\015\076\021" +
    "\071\022\057\023\055\025\074\026\075\027\066\030\056" +
    "\031\062\034\064\035\073\036\072\037\070\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\006" +
    "\014\120\041\113\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\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\006\014\120" +
    "\041\225\001\001\000\002\001\001\000\002\001\001\000" +
    "\006\014\120\041\217\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\002\001\001\000\010\014" +
    "\120\040\212\041\213\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\006\014\120\041\122\001\001" +
    "\000\002\001\001\000\006\014\120\041\160\001\001\000" +
    "\006\014\120\041\157\001\001\000\006\014\120\041\156" +
    "\001\001\000\006\014\120\041\155\001\001\000\006\014" +
    "\120\041\154\001\001\000\006\014\120\041\153\001\001" +
    "\000\006\014\120\041\152\001\001\000\006\014\120\041" +
    "\151\001\001\000\006\014\120\041\150\001\001\000\006" +
    "\014\120\041\147\001\001\000\006\014\120\041\146\001" +
    "\001\000\006\014\120\041\145\001\001\000\006\014\120" +
    "\041\144\001\001\000\006\014\120\041\143\001\001\000" +
    "\006\014\120\041\142\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\002\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
    "\001\001\000\006\014\120\041\162\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\006\014\120" +
    "\041\174\001\001\000\010\014\120\020\170\041\167\001" +
    "\001\000\002\001\001\000\002\001\001\000\006\014\120" +
    "\041\173\001\001\000\002\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\006\014\120\041\177" +
    "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\006\014\120\041\204\001\001" +
    "\000\002\001\001\000\002\001\001\000\006\014\120\041" +
    "\207\001\001\000\002\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\002\001\001\000\006\014" +
    "\120\041\216\001\001\000\002\001\001\000\002\001\001" +
    "\000\002\001\001\000\006\014\120\041\221\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\006\014\120\041\232\001\001" +
    "\000\002\001\001\000\006\014\120\041\234\001\001\000" +
    "\002\001\001\000\006\014\120\041\236\001\001\000\002" +
    "\001\001\000\002\001\001\000\006\014\120\041\241\001" +
    "\001\000\002\001\001\000\006\014\120\041\243\001\001" +
    "\000\002\001\001\000\006\014\120\041\245\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\006\014\120\041\253\001" +
    "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
    "\000\006\014\120\041\257\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\004" +
    "\011\264\001\001\000\002\001\001\000\002\001\001\000" +
    "\006\014\120\041\270\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\010\014\120\040\276\041\213" +
    "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\010\014\120" +
    "\040\304\041\213\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\004\011\311" +
    "\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" +
    "\006\014\120\041\323\001\001\000\002\001\001\000\002" +
    "\001\001\000\046\005\326\006\061\007\065\010\063\014" +
    "\067\015\076\021\071\022\057\023\055\025\074\026\075" +
    "\027\066\030\056\031\062\034\064\035\073\036\072\037" +
    "\070\001\001\000\044\006\247\007\065\010\063\014\067" +
    "\015\076\021\071\022\057\023\055\025\074\026\075\027" +
    "\066\030\056\031\062\034\064\035\073\036\072\037\070" +
    "\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\010\011\336\013\340\017\337\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\006\011\336" +
    "\013\346\001\001\000\002\001\001\000\046\005\344\006" +
    "\061\007\065\010\063\014\067\015\076\021\071\022\057" +
    "\023\055\025\074\026\075\027\066\030\056\031\062\034" +
    "\064\035\073\036\072\037\070\001\001\000\044\006\247" +
    "\007\065\010\063\014\067\015\076\021\071\022\057\023" +
    "\055\025\074\026\075\027\066\030\056\031\062\034\064" +
    "\035\073\036\072\037\070\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\002\001\001\000\006" +
    "\014\120\041\352\001\001\000\002\001\001\000\002\001" +
    "\001\000\006\024\357\033\356\001\001\000\002\001\001" +
    "\000\004\024\364\001\001\000\002\001\001\000\006\014" +
    "\120\041\361\001\001\000\002\001\001\000\046\005\363" +
    "\006\061\007\065\010\063\014\067\015\076\021\071\022" +
    "\057\023\055\025\074\026\075\027\066\030\056\031\062" +
    "\034\064\035\073\036\072\037\070\001\001\000\044\006" +
    "\247\007\065\010\063\014\067\015\076\021\071\022\057" +
    "\023\055\025\074\026\075\027\066\030\056\031\062\034" +
    "\064\035\073\036\072\037\070\001\001\000\002\001\001" +
    "\000\002\001\001\000\046\005\367\006\061\007\065\010" +
    "\063\014\067\015\076\021\071\022\057\023\055\025\074" +
    "\026\075\027\066\030\056\031\062\034\064\035\073\036" +
    "\072\037\070\001\001\000\044\006\247\007\065\010\063" +
    "\014\067\015\076\021\071\022\057\023\055\025\074\026" +
    "\075\027\066\030\056\031\062\034\064\035\073\036\072" +
    "\037\070\001\001\000\002\001\001\000\010\011\336\013" +
    "\340\017\372\001\001\000\002\001\001\000\002\001\001" +
    "\000\046\005\375\006\061\007\065\010\063\014\067\015" +
    "\076\021\071\022\057\023\055\025\074\026\075\027\066" +
    "\030\056\031\062\034\064\035\073\036\072\037\070\001" +
    "\001\000\044\006\247\007\065\010\063\014\067\015\076" +
    "\021\071\022\057\023\055\025\074\026\075\027\066\030" +
    "\056\031\062\034\064\035\073\036\072\037\070\001\001" +
    "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
    "\002\001\001\000\002\001\001\000\006\014\120\041\u0107" +
    "\001\001\000\006\014\120\041\u0105\001\001\000\002\001" +
    "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
    "\000\006\014\120\041\u010d\001\001\000\006\014\120\041" +
    "\u010b\001\001\000\002\001\001\000\002\001\001\000\002" +
    "\001\001\000\002\001\001\000\006\014\120\041\u0110\001" +
    "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
    "\000\002\001\001\000\002\001\001\000\006\014\120\041" +
    "\u0116\001\001\000\002\001\001\000\002\001\001\000\006" +
    "\014\120\041\u0119\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\006\014\120\041\u011d\001\001" +
    "\000\002\001\001\000\002\001\001\000\046\005\u0120\006" +
    "\061\007\065\010\063\014\067\015\076\021\071\022\057" +
    "\023\055\025\074\026\075\027\066\030\056\031\062\034" +
    "\064\035\073\036\072\037\070\001\001\000\044\006\247" +
    "\007\065\010\063\014\067\015\076\021\071\022\057\023" +
    "\055\025\074\026\075\027\066\030\056\031\062\034\064" +
    "\035\073\036\072\037\070\001\001\000\002\001\001\000" +
    "\004\022\u0123\001\001\000\002\001\001\000\046\005\u0125" +
    "\006\061\007\065\010\063\014\067\015\076\021\071\022" +
    "\057\023\055\025\074\026\075\027\066\030\056\031\062" +
    "\034\064\035\073\036\072\037\070\001\001\000\044\006" +
    "\247\007\065\010\063\014\067\015\076\021\071\022\057" +
    "\023\055\025\074\026\075\027\066\030\056\031\062\034" +
    "\064\035\073\036\072\037\070\001\001\000\002\001\001" +
    "\000\002\001\001\000\004\010\u012a\001\001\000\002\001" +
    "\001\000\006\014\120\041\u012b\001\001\000\002\001\001" +
    "\000\004\032\u012e\001\001\000\002\001\001\000\002\001" +
    "\001\000\002\001\001\000\046\005\u0131\006\061\007\065" +
    "\010\063\014\067\015\076\021\071\022\057\023\055\025" +
    "\074\026\075\027\066\030\056\031\062\034\064\035\073" +
    "\036\072\037\070\001\001\000\044\006\247\007\065\010" +
    "\063\014\067\015\076\021\071\022\057\023\055\025\074" +
    "\026\075\027\066\030\056\031\062\034\064\035\073\036" +
    "\072\037\070\001\001\000\002\001\001\000\002\001\001" +
    "\000\002\001\001\000\006\014\120\041\u0136\001\001\000" +
    "\002\001\001\000\006\014\120\041\u0138\001\001\000\002" +
    "\001\001\000\002\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$Parser$actions action_obj;

   /** Action encapsulation object initializer. */
   protected void init_actions()
    {
     action_obj = new CUP$Parser$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$Parser$do_action(act_num, parser, stack, top);
   }

   /** Indicates start state. */
   public int start_state() {return 0;}
   /** Indicates start production. */
   public int start_production() {return 0;}

   /** <code>EOF</code> Symbol index. */
   public int EOF_sym() {return 0;}

   /** <code>error</code> Symbol index. */
   public int error_sym() {return 1;}



    public LinkedList<Instruccion> AST;

    public void syntax_error(Symbol s){
        System.err.println("Error Sintáctico en la Línea " + s.left + " Columna " + s.right + ". No se
esperaba este componente: " +s.value);
        Errores.agregar("Sintáctico", "No se esperaba: " + s.value, s.left, s.right);
    }

    public void unrecovered_syntax_error(Symbol s) throws java.lang.Exception{
        System.err.println("Error sintáctico irrecuperable en la Línea " + s.left + " Columna " + s.right + ".
Componente " + s.value + " no reconocido.");
        Errores.agregar("Sintáctico Irrecuperable", "Componente no reconocido: " + s.value, s.left,
s.right);
    }


/** Cup generated class to encapsulate user supplied action code.*/
@SuppressWarnings({"rawtypes", "unchecked", "unused"})
class CUP$Parser$actions {
   private final Parser parser;

   /** Constructor */
   CUP$Parser$actions(Parser parser) {
    this.parser = parser;
   }

   /** Method 0 with the actual generated action code for actions 0 to 300. */
   public final java_cup.runtime.Symbol CUP$Parser$do_action_part00000000(
    int                        CUP$Parser$act_num,
    java_cup.runtime.lr_parser CUP$Parser$parser,
    java.util.Stack            CUP$Parser$stack,
    int                        CUP$Parser$top)
    throws java.lang.Exception
    {
     /* Symbol object for return from actions */
     java_cup.runtime.Symbol CUP$Parser$result;

     /* select the action based on the action number */
     switch (CUP$Parser$act_num)
        {
```

```
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 0: // $START ::= ini EOF
          {
            Object RESULT =null;
                        int start_valleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int start_valright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> start_val =
(LinkedList<Instruccion>)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).value;
                        RESULT = start_val;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("$START",0,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        /* ACCEPT */
        CUP$Parser$parser.done_parsing();
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 1: // ini ::= instrucciones_globales
          {
            LinkedList<Instruccion> RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        LinkedList<Instruccion> a = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         AST = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("ini",0,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 2: // instrucciones_globales ::= instrucciones_globales instruccion_global
          {
            LinkedList<Instruccion> RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> a = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         if(b!=null) a.add(b); RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instrucciones_globales",1,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 3: // instrucciones_globales ::= instruccion_global
          {
            LinkedList<Instruccion> RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         LinkedList<Instruccion> l = new LinkedList<>(); if(a!=null) l.add(a); RESULT = l;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instrucciones_globales",1,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 4: // instruccion_global ::= declaracion
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 5: // instruccion_global ::= declaracion_vector
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 6: // instruccion_global ::= declaracion_lista
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 7: // instruccion_global ::= funcion
         {
           Instruccion RESULT =null;
                     int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                     int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                     Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                      RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 8: // instruccion_global ::= start_with
         {
           Instruccion RESULT =null;
                     int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                     int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                     Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                      RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 9: // instruccion_global ::= asignacion
         {
           Instruccion RESULT =null;
                     int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                     int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                     Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                      RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 10: // instruccion_global ::= modificacion_vector
         {
           Instruccion RESULT =null;
                     int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                     int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                     Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                      RESULT = a;
```

```
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 11: // instruccion_global ::= impresion
        {
          Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 12: // instruccion_global ::= sentencia_if
        {
          Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 13: // instruccion_global ::= sentencia_switch
        {
          Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 14: // instruccion_global ::= sentencia_while
        {
          Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
```

```
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 15: // instruccion_global ::= sentencia_do_while
        {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 16: // instruccion_global ::= sentencia_for
        {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 17: // instruccion_global ::= sentencia_break
        {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 18: // instruccion_global ::= sentencia_continue
        {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 19: // instruccion_global ::= llamada_funcion PTCOMA
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 20: // instruccion_global ::= metodo_lista PTCOMA
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion_global",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 21: // start_with ::= START_WITH ID PAR_A PAR_C PTCOMA
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                         RESULT = new StartWith(id, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("start_with",12,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 22: // funcion ::= tipo ID PAR_A lista_parametros PAR_C LLAVE_A instrucciones LLAVE_C
          {
            Instruccion RESULT =null;
                        int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).left;
                        int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).right;
                        String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-7)).value;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int pleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int pright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        LinkedList<Instruccion> p = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int bloqueleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int bloqueright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> bloque =
(LinkedList<Instruccion>)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).value;
                         RESULT = new Funcion(t, id, p, bloque, tleft, tright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("funcion",8,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 23: // funcion ::= VOID ID PAR_A lista_parametros PAR_C LLAVE_A instrucciones LLAVE_C
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int pleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int pright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
```

```
                         LinkedList<Instruccion> p = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                         int bloqueleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                         int bloqueright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                         LinkedList<Instruccion> bloque =
(LinkedList<Instruccion>)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).value;
                          RESULT = new Funcion("void", id, p, bloque, idleft, idright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("funcion",8,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 24: // lista_parametros ::= lista_parametros COMA parametro
            {
              LinkedList<Instruccion> RESULT =null;
                         int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
2)).left;
                         int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                         LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                         int pleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                         int pright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                         Instruccion p = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                          l.add(p); RESULT = l;
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_parametros",13,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 25: // lista_parametros ::= parametro
            {
              LinkedList<Instruccion> RESULT =null;
                         int pleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                         int pright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                         Instruccion p = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                          LinkedList<Instruccion> l = new LinkedList<>(); l.add(p); RESULT = l;
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_parametros",13,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 26: // lista_parametros ::=
            {
              LinkedList<Instruccion> RESULT =null;
                          RESULT = new LinkedList<Instruccion>();
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_parametros",13,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 27: // parametro ::= tipo ID
          {
            Instruccion RESULT =null;
                        int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                        int idleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int idright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        String id = (String)((java_cup.runtime.Symbol) CUP$Parser$stack.peek()).value;
                         RESULT = new Declaracion(id, t, null, tleft, tright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("parametro",9,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 28: // instrucciones ::= instrucciones instruccion
          {
            LinkedList<Instruccion> RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> a = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         if(b!=null) a.add(b); RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instrucciones",3,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 29: // instrucciones ::= instruccion
          {
            LinkedList<Instruccion> RESULT =null;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         LinkedList<Instruccion> l = new LinkedList<>(); if(b!=null) l.add(b); RESULT = l;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instrucciones",3,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 30: // instruccion ::= declaracion
         {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 31: // instruccion ::= declaracion_vector
         {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 32: // instruccion ::= declaracion_lista
         {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 33: // instruccion ::= asignacion
         {
          Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 34: // instruccion ::= modificacion_vector
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 35: // instruccion ::= impresion
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 36: // instruccion ::= sentencia_if
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 37: // instruccion ::= sentencia_switch
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 38: // instruccion ::= sentencia_while
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 39: // instruccion ::= sentencia_do_while
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 40: // instruccion ::= sentencia_for
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 41: // instruccion ::= sentencia_break
          {
            Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 42: // instruccion ::= sentencia_continue
          {
            Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 43: // instruccion ::= sentencia_return
          {
            Instruccion RESULT =null;
                    int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     RESULT = a;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 44: // instruccion ::= llamada_funcion PTCOMA
          {
            Instruccion RESULT =null;
                    int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                    int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                    Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                     RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 45: // instruccion ::= metodo_lista PTCOMA
        {
          Instruccion RESULT =null;
                int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                 RESULT = a;
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("instruccion",4,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 46: // declaracion ::= VAR ID DOSPTOS tipo IGUAL expresion PTCOMA
        {
          Instruccion RESULT =null;
                int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).left;
                int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).right;
                String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-5)).value;
                int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                 RESULT = new Declaracion(id, t, e, idleft, idright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("declaracion",5,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 47: // declaracion ::= VAR ID DOSPTOS tipo PTCOMA
        {
          Instruccion RESULT =null;
```

```
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                        int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Declaracion(id, t, null, idleft, idright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("declaracion",5,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 48: // tipo ::= INT
          {
           String RESULT =null;
                         RESULT = "int";
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("tipo",7,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 49: // tipo ::= DOUBLE
          {
           String RESULT =null;
                         RESULT = "double";
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("tipo",7,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 50: // tipo ::= BOOL
          {
           String RESULT =null;
                         RESULT = "bool";
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("tipo",7,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 51: // tipo ::= CHAR
          {
           String RESULT =null;
                         RESULT = "char";
```

```
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("tipo",7,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 52: // tipo ::= STRING
          {
            String RESULT =null;
                        RESULT = "string";
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("tipo",7,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 53: // asignacion ::= ID IGUAL expresion PTCOMA
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                        int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                        RESULT = new Asignacion(id, e, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("asignacion",6,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 54: // asignacion ::= ID MASMAS PTCOMA
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        RESULT = new Asignacion(id, new Aritmetica(new AccesoVar(id, idleft, idright), new
Dato("1", "int", 0, 0), Operacion.Tipo_Operacion.SUMA, idleft, idright), idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("asignacion",6,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 55: // asignacion ::= ID MENOSMENOS PTCOMA
         {
          Instruccion RESULT =null;
                    int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                    int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                    String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                     RESULT = new Asignacion(id, new Aritmetica(new AccesoVar(id, idleft, idright), new
Dato("1", "int", 0, 0), Operacion.Tipo_Operacion.RESTA, idleft, idright), idleft, idright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("asignacion",6,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 56: // declaracion_vector ::= VAR ID DOSPTOS tipo COR_IZQ COR_DER IGUAL COR_IZQ
lista_valores COR_DER PTCOMA
         {
          Instruccion RESULT =null;
                    int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)).left;
                    int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)).right;
                    String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-9)).value;
                    int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).left;
                    int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).right;
                    String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-7)).value;
                    int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
2)).left;
                    int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                    LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                     RESULT = new DeclaracionVector(id, t, 1, l, idleft, idright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("declaracion_vector",26,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 57: // declaracion_vector ::= VAR ID DOSPTOS tipo COR_IZQ COR_DER COR_IZQ COR_DER
IGUAL COR_IZQ lista_valores COR_DER PTCOMA
         {
          Instruccion RESULT =null;
                    int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-11)).left;
```

```
                    int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-11)).right;
                    String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-11)).value;
                    int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)).left;
                    int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)).right;
                    String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-9)).value;
                    int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
2)).left;
                    int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                    LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                     RESULT = new DeclaracionVector(id, t, 2, l, idleft, idright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("declaracion_vector",26,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-12)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 58: // lista_valores ::= lista_valores COMA expresion
            {
              LinkedList<Instruccion> RESULT =null;
                    int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
2)).left;
                    int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                    LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                    int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     l.add(e); RESULT = l;
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_valores",30,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 59: // lista_valores ::= expresion
            {
              LinkedList<Instruccion> RESULT =null;
                    int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                     LinkedList<Instruccion> l = new LinkedList<>(); l.add(e); RESULT = l;
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_valores",30,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
```

```
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 60: // modificacion_vector ::= ID COR_IZQ expresion COR_DER IGUAL expresion PTCOMA
          {
            Instruccion RESULT =null;
                        int idleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int idright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int indleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int indright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion ind = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int valleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int valright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion val = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new ModificacionVector(id, ind, null, val, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("modificacion_vector",27, 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)), 
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 61: // modificacion_vector ::= ID COR_IZQ expresion COR_DER COR_IZQ expresion 
COR_DER IGUAL expresion PTCOMA
          {
            Instruccion RESULT =null;
                        int idleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)).left;
                        int idright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-9)).value;
                        int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
7)).left;
                        int iright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).right;
                        Instruccion i = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-7)).value;
                        int jleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
4)).left;
                        int jright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion j = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int valleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
```

```
                        int valright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion val = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new ModificacionVector(id, i, j, val, idleft, idright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("modificacion_vector",27,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . .*/
        case 62: // declaracion_lista ::= LIST MENOR tipo MAYOR ID IGUAL NEW LIST PAR_A PAR_C
PTCOMA
          {
           Instruccion RESULT =null;
                        int tleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).left;
                        int tright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).right;
                        String t = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-8)).value;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                         RESULT = new DeclaracionLista(id, t, idleft, idright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("declaracion_lista",28,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . .*/
        case 63: // metodo_lista ::= ID PUNTO APPEND PAR_A expresion PAR_C
          {
           Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-5)).value;
                        int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new MetodoLista(id, "append", e, idleft, idright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("metodo_lista",29,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 64: // metodo_lista ::= ID PUNTO REMOVE PAR_A expresion PAR_C
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-5)).value;
                        int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new MetodoLista(id, "remove", e, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("metodo_lista",29,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 65: // impresion ::= PRINT PAR_A expresion PAR_C PTCOMA
          {
            Instruccion RESULT =null;
                        int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                         RESULT = new Print(e, false, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("impresion",15,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 66: // impresion ::= PRINTLN PAR_A expresion PAR_C PTCOMA
          {
            Instruccion RESULT =null;
                        int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                         RESULT = new Print(e, true, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("impresion",15,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 67: // sentencia_if ::= IF PAR_A expresion PAR_C LLAVE_A instrucciones LLAVE_C
          {
            Instruccion RESULT =null;
                        int condleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int condright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion cond = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int i1left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int i1right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> i1 = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new If(cond, i1, null, condleft, condright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_if",16,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 68: // sentencia_if ::= IF PAR_A expresion PAR_C LLAVE_A instrucciones LLAVE_C ELSE
LLAVE_A instrucciones LLAVE_C
          {
            Instruccion RESULT =null;
                        int condleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).left;
                        int condright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).right;
                        Instruccion cond = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-8)).value;
                        int i1left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).left;
                        int i1right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).right;
                        LinkedList<Instruccion> i1 = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-5)).value;
                        int i2left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int i2right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> i2 = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new If(cond, i1, i2, condleft, condright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_if",16,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 69: // sentencia_if ::= IF PAR_A expresion PAR_C LLAVE_A instrucciones LLAVE_C ELSE
sentencia_if
```

```
          {
            Instruccion RESULT =null;
                        int condleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int condright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        Instruccion cond = (Instruccion)((java_cup.runtime.Symbol) 
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int i1left = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                        int i1right = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                        LinkedList<Instruccion> i1 = (LinkedList<Instruccion>)((java_cup.runtime.Symbol) 
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                        int i2left = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int i2right = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion i2 = (Instruccion)((java_cup.runtime.Symbol) 
CUP$Parser$stack.peek()).value;
                         LinkedList<Instruccion> l = new LinkedList<>(); l.add(i2); RESULT = new If(cond, i1, l, 
condleft, condright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_if",16, 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)), 
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 70: // sentencia_switch ::= SWITCH PAR_A expresion PAR_C LLAVE_A lista_casos LLAVE_C
          {
            Instruccion RESULT =null;
                        int eleft = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int eright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol) 
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int lright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol) 
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Switch(e, l, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_switch",17, 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)), 
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 71: // lista_casos ::= lista_casos caso
          {
            LinkedList<Instruccion> RESULT =null;
                        int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int lright = 
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
```

```
                         LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         int cleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                         int cright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                         Instruccion c = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                          if(c!=null) l.add(c); RESULT = l;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_casos",25,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 72: // lista_casos ::= caso
          {
            LinkedList<Instruccion> RESULT =null;
                         int cleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                         int cright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                         Instruccion c = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                          LinkedList<Instruccion> l = new LinkedList<>(); if(c!=null) l.add(c); RESULT = l;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_casos",25,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 73: // caso ::= CASE expresion DOSPTOS instrucciones
          {
            Instruccion RESULT =null;
                         int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                         int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                         Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                         int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                         int iright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                         LinkedList<Instruccion> i = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                          RESULT = new Caso(e, i, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("caso",18,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 74: // caso ::= DEFAULT DOSPTOS instrucciones
          {
            Instruccion RESULT =null;
                         int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                         int iright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                         LinkedList<Instruccion> i = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
```

```
                        RESULT = new Caso(null, i, ileft, iright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("caso",18,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 75: // sentencia_while ::= WHILE PAR_A expresion PAR_C LLAVE_A instrucciones LLAVE_C
            {
              Instruccion RESULT =null;
                        int condleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int condright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion cond = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int iright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> i = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new While(cond, i, condleft, condright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_while",19,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 76: // sentencia_do_while ::= DO LLAVE_A instrucciones LLAVE_C WHILE PAR_A expresion
PAR_C PTCOMA
            {
              Instruccion RESULT =null;
                        int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
6)).left;
                        int iright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        LinkedList<Instruccion> i = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int condleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int condright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion cond = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                         RESULT = new DoWhile(cond, i, condleft, condright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_do_while",20,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 77: // actualizacion_for ::= ID IGUAL expresion
            {
```

```
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Asignacion(id, e, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("actualizacion_for",24,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 78: // actualizacion_for ::= ID MASMAS
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Asignacion(id, new Aritmetica(new AccesoVar(id,0,0), new
Dato("1","int",0,0), Operacion.Tipo_Operacion.SUMA, 0,0), idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("actualizacion_for",24,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 79: // actualizacion_for ::= ID MENOSMENOS
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Asignacion(id, new Aritmetica(new AccesoVar(id,0,0), new
Dato("1","int",0,0), Operacion.Tipo_Operacion.RESTA, 0,0), idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("actualizacion_for",24,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 80: // sentencia_for ::= FOR PAR_A asignacion expresion PTCOMA actualizacion_for PAR_C
LLAVE_A instrucciones LLAVE_C
```

```
          {
            Instruccion RESULT =null;
                        int inileft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).left;
                        int iniright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).right;
                        Instruccion ini = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-7)).value;
                        int condleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int condright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        Instruccion cond = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int incleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).left;
                        int incright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion inc = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
                        int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int iright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> i = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new For(ini, cond, inc, i, inileft, iniright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_for",21,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 81: // sentencia_break ::= BREAK PTCOMA
          {
            Instruccion RESULT =null;
                        int bleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int bright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Object b = (Object)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Break(bleft, bright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_break",22,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 82: // sentencia_continue ::= CONTINUE PTCOMA
          {
            Instruccion RESULT =null;
                        int cleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
```

```
                        int cright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Object c = (Object)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Continue(cleft, cright);
             CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_continue",23,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
           }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 83: // sentencia_return ::= RETURN expresion PTCOMA
            {
             Instruccion RESULT =null;
                        int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new Return(e, eleft, eright);
             CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_return",11,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
           }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 84: // sentencia_return ::= RETURN PTCOMA
            {
             Instruccion RESULT =null;
                         RESULT = new Return(null, 0, 0);
             CUP$Parser$result = parser.getSymbolFactory().newSymbol("sentencia_return",11,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
           }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 85: // expresion ::= MENOS expresion
            {
             Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Aritmetica(a, Operacion.Tipo_Operacion.NEGACION, aleft, aright);
             CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
           }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 86: // expresion ::= PAR_A INT PAR_C expresion
            {
```

```
            Instruccion RESULT =null;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Casteo("int", e, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 87: // expresion ::= PAR_A DOUBLE PAR_C expresion
          {
            Instruccion RESULT =null;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Casteo("double", e, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 88: // expresion ::= PAR_A CHAR PAR_C expresion
          {
            Instruccion RESULT =null;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Casteo("char", e, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 89: // expresion ::= PAR_A STRING PAR_C expresion
          {
            Instruccion RESULT =null;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Casteo("string", e, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 90: // expresion ::= PAR_A BOOL PAR_C expresion
          {
            Instruccion RESULT =null;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Casteo("bool", e, eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 91: // expresion ::= expresion MAS expresion
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Aritmetica(a, b, Operacion.Tipo_Operacion.SUMA, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 92: // expresion ::= expresion MENOS expresion
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Aritmetica(a, b, Operacion.Tipo_Operacion.RESTA, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;
```

```
          /*. . . . . . . . . . . . . . . . . . . .*/
          case 93: // expresion ::= expresion POR expresion
            {
              Instruccion RESULT =null;
                              int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                              int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                              Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                              int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                              int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                              Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                               RESULT = new Aritmetica(a, b, Operacion.Tipo_Operacion.MULTIPLICACION, aleft,
aright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 94: // expresion ::= expresion DIV expresion
            {
              Instruccion RESULT =null;
                              int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                              int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                              Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                              int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                              int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                              Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                               RESULT = new Aritmetica(a, b, Operacion.Tipo_Operacion.DIVISION, aleft, aright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 95: // expresion ::= expresion POTENCIA expresion
            {
              Instruccion RESULT =null;
                              int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                              int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                              Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                              int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                              int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                              Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
```

```
                        RESULT = new Aritmetica(a, b, Operacion.Tipo_Operacion.POTENCIA, aleft, aright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 96: // expresion ::= expresion MOD expresion
            {
              Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                        RESULT = new Aritmetica(a, b, Operacion.Tipo_Operacion.MODULO, aleft, aright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 97: // expresion ::= expresion IGUALIGUAL expresion
            {
              Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                        RESULT = new Logica(a, b, Operacion.Tipo_Operacion.IGUALIGUAL, aleft, aright);
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 98: // expresion ::= expresion DIFERENTE expresion
            {
              Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
```

```
		Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
		int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
		int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
		Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
		 RESULT = new Logica(a, b, Operacion.Tipo_Operacion.DIFERENTE, aleft, aright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 99: // expresion ::= expresion MENOR expresion
        {
          Instruccion RESULT =null;
		int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
		int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
		Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
		int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
		int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
		Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
		 RESULT = new Logica(a, b, Operacion.Tipo_Operacion.MENOR, aleft, aright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 100: // expresion ::= expresion MAYOR expresion
        {
          Instruccion RESULT =null;
		int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
		int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
		Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
		int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
		int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
		Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
		 RESULT = new Logica(a, b, Operacion.Tipo_Operacion.MAYOR, aleft, aright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 101: // expresion ::= expresion MENORIGUAL expresion
```

```
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Logica(a, b, Operacion.Tipo_Operacion.MENORIGUAL, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 102: // expresion ::= expresion MAYORIGUAL expresion
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Logica(a, b, Operacion.Tipo_Operacion.MAYORIGUAL, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 103: // expresion ::= expresion AND expresion
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Logica(a, b, Operacion.Tipo_Operacion.AND, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 104: // expresion ::= expresion OR expresion
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Logica(a, b, Operacion.Tipo_Operacion.OR, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 105: // expresion ::= expresion XOR expresion
          {
            Instruccion RESULT =null;
                        int aleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int aright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int bleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int bright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion b = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Logica(a, b, Operacion.Tipo_Operacion.XOR, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 106: // expresion ::= NOT expresion
          {
            Instruccion RESULT =null;
                        int aleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int aright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion a = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = new Logica(a, Operacion.Tipo_Operacion.NOT, aleft, aright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 107: // expresion ::= PAR_A expresion PAR_C
          {
            Instruccion RESULT =null;
                    int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                    int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                    Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                     RESULT = e;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 108: // expresion ::= ID
          {
            Instruccion RESULT =null;
                    int idleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int idright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    String id = (String)((java_cup.runtime.Symbol) CUP$Parser$stack.peek()).value;
                     RESULT = new AccesoVar(id, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 109: // expresion ::= ENTERO
          {
            Instruccion RESULT =null;
                    int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    String e = (String)((java_cup.runtime.Symbol) CUP$Parser$stack.peek()).value;
                     RESULT = new Dato(e, "int", eleft, eright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 110: // expresion ::= DECIMAL
          {
            Instruccion RESULT =null;
                    int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    String e = (String)((java_cup.runtime.Symbol) CUP$Parser$stack.peek()).value;
                     RESULT = new Dato(e, "double", eleft, eright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 111: // expresion ::= LIT_STRING
        {
          Instruccion RESULT =null;
                    int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    String e = (String)((java_cup.runtime.Symbol) CUP$Parser$stack.peek()).value;
                     RESULT = new Dato(e, "string", eleft, eright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 112: // expresion ::= LIT_CHAR
        {
          Instruccion RESULT =null;
                    int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                    int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                    String e = (String)((java_cup.runtime.Symbol) CUP$Parser$stack.peek()).value;
                     RESULT = new Dato(e, "char", eleft, eright);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 113: // expresion ::= TRUE
        {
          Instruccion RESULT =null;
                     RESULT = new Dato("true", "bool", 0, 0);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 114: // expresion ::= FALSE
        {
          Instruccion RESULT =null;
                     RESULT = new Dato("false", "bool", 0, 0);
          CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;
```

```
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 115: // expresion ::= COR_IZQ lista_valores COR_DER
          {
            Instruccion RESULT =null;
                        int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new VectorLiteral(l, lleft, lright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 116: // expresion ::= ID COR_IZQ expresion COR_DER
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                        int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int iright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        Instruccion i = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new AccesoVector(id, i, null, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 117: // expresion ::= ID COR_IZQ expresion COR_DER COR_IZQ expresion COR_DER
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-6)).value;
                        int ileft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
4)).left;
                        int iright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)).right;
                        Instruccion i = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-4)).value;
```

```
                int jleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                int jright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Instruccion j = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                 RESULT = new AccesoVector(id, i, j, idleft, idright);
        CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-6)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
      }
     return CUP$Parser$result;

     /*. . . . . . . . . . . . . . . . . . . .*/
     case 118: // expresion ::= ROUND PAR_A expresion PAR_C
      {
        Instruccion RESULT =null;
                int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                 RESULT = new Nativa(e, "round", eleft, eright);
        CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
      }
     return CUP$Parser$result;

     /*. . . . . . . . . . . . . . . . . . . .*/
     case 119: // expresion ::= LENGTH PAR_A expresion PAR_C
      {
        Instruccion RESULT =null;
                int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                 RESULT = new Nativa(e, "length", eleft, eright);
        CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
      }
     return CUP$Parser$result;

     /*. . . . . . . . . . . . . . . . . . . .*/
     case 120: // expresion ::= TOSTRING PAR_A expresion PAR_C
      {
        Instruccion RESULT =null;
                int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
```

```
                     RESULT = new Nativa(e, "toString", eleft, eright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 121: // expresion ::= ID PUNTO FIND PAR_A expresion PAR_C
          {
            Instruccion RESULT =null;
                    int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).left;
                    int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).right;
                    String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-5)).value;
                    int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                    int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                    Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                     RESULT = new FindVector(id, e, idleft, idright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 122: // expresion ::= ID PUNTO REMOVE PAR_A expresion PAR_C
          {
            Instruccion RESULT =null;
                    int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).left;
                    int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)).right;
                    String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-5)).value;
                    int eleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                    int eright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                    Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                     RESULT = new MetodoLista(id, "remove", e, idleft, idright);
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 123: // expresion ::= llamada_funcion
          {
            Instruccion RESULT =null;
```

```
                        int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int lright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion l = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         RESULT = l;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("expresion",31,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 124: // llamada_funcion ::= ID PAR_A lista_argumentos PAR_C
          {
            Instruccion RESULT =null;
                        int idleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                        int idright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                        String id = (String)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                        int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
1)).left;
                        int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                        LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                         RESULT = new LlamadaFuncion(id, l, idleft, idright);
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("llamada_funcion",10,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 125: // lista_argumentos ::= lista_argumentos COMA expresion
          {
            LinkedList<Instruccion> RESULT =null;
                        int lleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-
2)).left;
                        int lright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        LinkedList<Instruccion> l = (LinkedList<Instruccion>)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         l.add(e); RESULT = l;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_argumentos",14,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 126: // lista_argumentos ::= expresion
```

```
          {
            LinkedList<Instruccion> RESULT =null;
                        int eleft = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).left;
                        int eright = ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()).right;
                        Instruccion e = (Instruccion)((java_cup.runtime.Symbol)
CUP$Parser$stack.peek()).value;
                         LinkedList<Instruccion> l = new LinkedList<>(); l.add(e); RESULT = l;
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_argumentos",14,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 127: // lista_argumentos ::=
          {
            LinkedList<Instruccion> RESULT =null;
                         RESULT = new LinkedList<Instruccion>();
            CUP$Parser$result = parser.getSymbolFactory().newSymbol("lista_argumentos",14,
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /* . . . . . .*/
        default:
          throw new Exception(
             "Invalid action number "+CUP$Parser$act_num+"found in internal parse table");

      }
    } /* end of method */

  /** Method splitting the generated action code into several parts. */
  public final java_cup.runtime.Symbol CUP$Parser$do_action(
    int                        CUP$Parser$act_num,
    java_cup.runtime.lr_parser CUP$Parser$parser,
    java.util.Stack            CUP$Parser$stack,
    int                        CUP$Parser$top)
    throws java.lang.Exception
    {
            return CUP$Parser$do_action_part00000000(
                        CUP$Parser$act_num,
                        CUP$Parser$parser,
                        CUP$Parser$stack,
                        CUP$Parser$top);
    }
}

}
```