```java
/* Generated By:JavaCC: Do not edit this line. ParserTokenManager.java */
import AST.*;

/** Token Manager. */
public class ParserTokenManager implements ParserConstants
{

  /** Debug output. */
  public static  java.io.PrintStream debugStream = System.out;
  /** Set debug output. */
  public static  void setDebugStream(java.io.PrintStream ds) { debugStream = ds; }
private static final int jjStopStringLiteralDfa_0(int pos, long active0)
{
   switch (pos)
   {
      case 0:
         if ((active0 & 0x7480fe0L) != 0L)
         {
            jjmatchedKind = 28;
            return 1;
         }
         return -1;
      case 1:
         if ((active0 & 0x54809e0L) != 0L)
         {
            jjmatchedKind = 28;
            jjmatchedPos = 1;
            return 1;
         }
         if ((active0 & 0x2000600L) != 0L)
            return 1;
         return -1;
      case 2:
         if ((active0 & 0x9e0L) != 0L)
         {
            jjmatchedKind = 28;
            jjmatchedPos = 2;
            return 1;
         }
         if ((active0 & 0x5480000L) != 0L)
            return 1;
         return -1;
      case 3:
         if ((active0 & 0x840L) != 0L)
         {
            jjmatchedKind = 28;
            jjmatchedPos = 3;
            return 1;
         }
         if ((active0 & 0x1a0L) != 0L)
            return 1;
         return -1;
      default :
         return -1;
   }
}
private static final int jjStartNfa_0(int pos, long active0)
{
   return jjMoveNfa_0(jjStopStringLiteralDfa_0(pos, active0), pos + 1);
}
static private int jjStopAtPos(int pos, int kind)
{
```

```
 67    jjmatchedKind = kind;
 68    jjmatchedPos = pos;
 69    return pos + 1;
 70 }
 71 static private int jjMoveStringLiteralDfa0_0()
 72 {
 73    switch(curChar)
 74    {
 75       case 33:
 76          return jjStopAtPos(0, 18);
 77       case 40:
 78          return jjStopAtPos(0, 34);
 79       case 41:
 80          return jjStopAtPos(0, 35);
 81       case 42:
 82          return jjStopAtPos(0, 32);
 83       case 43:
 84          return jjStopAtPos(0, 30);
 85       case 44:
 86          return jjStopAtPos(0, 20);
 87       case 45:
 88          return jjStopAtPos(0, 31);
 89       case 47:
 90          return jjStopAtPos(0, 33);
 91       case 58:
 92          return jjMoveStringLiteralDfa1_0(0x20000L);
 93       case 59:
 94          jjmatchedKind = 21;
 95          return jjMoveStringLiteralDfa1_0(0x1000000000L);
 96       case 60:
 97          jjmatchedKind = 14;
 98          return jjMoveStringLiteralDfa1_0(0x1000L);
 99       case 61:
100          jjmatchedKind = 27;
101          return jjMoveStringLiteralDfa1_0(0x810000L);
102       case 62:
103          jjmatchedKind = 15;
104          return jjMoveStringLiteralDfa1_0(0x2000L);
105       case 100:
106          return jjMoveStringLiteralDfa1_0(0x400L);
107       case 101:
108          return jjMoveStringLiteralDfa1_0(0x4000080L);
109       case 102:
110          return jjMoveStringLiteralDfa1_0(0x400040L);
111       case 105:
112          return jjMoveStringLiteralDfa1_0(0x2000200L);
113       case 108:
114          return jjMoveStringLiteralDfa1_0(0x1000000L);
115       case 110:
116          return jjMoveStringLiteralDfa1_0(0x80000L);
117       case 116:
118          return jjMoveStringLiteralDfa1_0(0x120L);
119       case 119:
120          return jjMoveStringLiteralDfa1_0(0x800L);
121       default :
122          return jjMoveNfa_0(0, 0);
123    }
124 }
125 static private int jjMoveStringLiteralDfa1_0(long active0)
126 {
127    try { curChar = input_stream.readChar(); }
128    catch(java.io.IOException e) {
```

```
129       jjStopStringLiteralDfa_0(0, active0);
130       return 1;
131    }
132    switch(curChar)
133    {
134       case 59:
135          if ((active0 & 0x1000000000L) != 0L)
136             return jjStopAtPos(1, 36);
137          break;
138       case 61:
139          if ((active0 & 0x1000L) != 0L)
140             return jjStopAtPos(1, 12);
141          else if ((active0 & 0x2000L) != 0L)
142             return jjStopAtPos(1, 13);
143          else if ((active0 & 0x10000L) != 0L)
144             return jjStopAtPos(1, 16);
145          else if ((active0 & 0x20000L) != 0L)
146             return jjStopAtPos(1, 17);
147          break;
148       case 62:
149          if ((active0 & 0x800000L) != 0L)
150             return jjStopAtPos(1, 23);
151          break;
152       case 97:
153          return jjMoveStringLiteralDfa2_0(active0, 0x40L);
154       case 101:
155          return jjMoveStringLiteralDfa2_0(active0, 0x1080000L);
156       case 102:
157          if ((active0 & 0x200L) != 0L)
158             return jjStartNfaWithStates_0(1, 9, 1);
159          break;
160       case 104:
161          return jjMoveStringLiteralDfa2_0(active0, 0x900L);
162       case 108:
163          return jjMoveStringLiteralDfa2_0(active0, 0x80L);
164       case 110:
165          if ((active0 & 0x2000000L) != 0L)
166             return jjStartNfaWithStates_0(1, 25, 1);
167          return jjMoveStringLiteralDfa2_0(active0, 0x4000000L);
168       case 111:
169          if ((active0 & 0x400L) != 0L)
170             return jjStartNfaWithStates_0(1, 10, 1);
171          break;
172       case 114:
173          return jjMoveStringLiteralDfa2_0(active0, 0x20L);
174       case 117:
175          return jjMoveStringLiteralDfa2_0(active0, 0x400000L);
176       default :
177          break;
178    }
179    return jjStartNfa_0(0, active0);
180 }
181 static private int jjMoveStringLiteralDfa2_0(long old0, long active0)
182 {
183    if (((active0 &= old0)) == 0L)
184       return jjStartNfa_0(0, old0);
185    try { curChar = input_stream.readChar(); }
186    catch(java.io.IOException e) {
187       jjStopStringLiteralDfa_0(1, active0);
188       return 2;
189    }
190    switch(curChar)
```

```
191    {
192       case 100:
193          if ((active0 & 0x4000000L) != 0L)
194             return jjStartNfaWithStates_0(2, 26, 1);
195          break;
196       case 101:
197          return jjMoveStringLiteralDfa3_0(active0, 0x100L);
198       case 105:
199          return jjMoveStringLiteralDfa3_0(active0, 0x800L);
200       case 108:
201          return jjMoveStringLiteralDfa3_0(active0, 0x40L);
202       case 110:
203          if ((active0 & 0x400000L) != 0L)
204             return jjStartNfaWithStates_0(2, 22, 1);
205          break;
206       case 115:
207          return jjMoveStringLiteralDfa3_0(active0, 0x80L);
208       case 116:
209          if ((active0 & 0x1000000L) != 0L)
210             return jjStartNfaWithStates_0(2, 24, 1);
211          break;
212       case 117:
213          return jjMoveStringLiteralDfa3_0(active0, 0x20L);
214       case 119:
215          if ((active0 & 0x80000L) != 0L)
216             return jjStartNfaWithStates_0(2, 19, 1);
217          break;
218       default :
219          break;
220    }
221    return jjStartNfa_0(1, active0);
222 }
223 static private int jjMoveStringLiteralDfa3_0(long old0, long active0)
224 {
225    if (((active0 &= old0)) == 0L)
226       return jjStartNfa_0(1, old0);
227    try { curChar = input_stream.readChar(); }
228    catch(java.io.IOException e) {
229       jjStopStringLiteralDfa_0(2, active0);
230       return 3;
231    }
232    switch(curChar)
233    {
234       case 101:
235          if ((active0 & 0x20L) != 0L)
236             return jjStartNfaWithStates_0(3, 5, 1);
237          else if ((active0 & 0x80L) != 0L)
238             return jjStartNfaWithStates_0(3, 7, 1);
239          break;
240       case 108:
241          return jjMoveStringLiteralDfa4_0(active0, 0x800L);
242       case 110:
243          if ((active0 & 0x100L) != 0L)
244             return jjStartNfaWithStates_0(3, 8, 1);
245          break;
246       case 115:
247          return jjMoveStringLiteralDfa4_0(active0, 0x40L);
248       default :
249          break;
250    }
251    return jjStartNfa_0(2, active0);
252 }
```

```
253 static private int jjMoveStringLiteralDfa4_0(long old0, long active0)
254 {
255    if (((active0 &= old0)) == 0L)
256       return jjStartNfa_0(2, old0);
257    try { curChar = input_stream.readChar(); }
258    catch(java.io.IOException e) {
259       jjStopStringLiteralDfa_0(3, active0);
260       return 4;
261    }
262    switch(curChar)
263    {
264       case 101:
265          if ((active0 & 0x40L) != 0L)
266             return jjStartNfaWithStates_0(4, 6, 1);
267          else if ((active0 & 0x800L) != 0L)
268             return jjStartNfaWithStates_0(4, 11, 1);
269          break;
270       default :
271          break;
272    }
273    return jjStartNfa_0(3, active0);
274 }
275 static private int jjStartNfaWithStates_0(int pos, int kind, int state)
276 {
277    jjmatchedKind = kind;
278    jjmatchedPos = pos;
279    try { curChar = input_stream.readChar(); }
280    catch(java.io.IOException e) { return pos + 1; }
281    return jjMoveNfa_0(state, pos + 1);
282 }
283 static private int jjMoveNfa_0(int startState, int curPos)
284 {
285    int startsAt = 0;
286    jjnewStateCnt = 3;
287    int i = 1;
288    jjstateSet[0] = startState;
289    int kind = 0x7fffffff;
290    for (;;)
291    {
292       if (++jjround == 0x7fffffff)
293          ReInitRounds();
294       if (curChar < 64)
295       {
296          long l = 1L << curChar;
297          do
298          {
299             switch(jjstateSet[--i])
300             {
301                case 0:
302                case 2:
303                   if ((0x3ff000000000000L & l) == 0L)
304                      break;
305                   if (kind > 29)
306                      kind = 29;
307                   jjCheckNAdd(2);
308                   break;
309                case 1:
310                   if ((0x3ff000000000000L & l) == 0L)
311                      break;
312                   if (kind > 28)
313                      kind = 28;
314                   jjstateSet[jjnewStateCnt++] = 1;
```

```
315                   break;
316                default : break;
317             }
318          } while(i != startsAt);
319       }
320       else if (curChar < 128)
321       {
322          long l = 1L << (curChar & 077);
323          do
324          {
325             switch(jjstateSet[--i])
326             {
327                case 0:
328                case 1:
329                   if ((0x7fffffe07fffffeL & l) == 0L)
330                      break;
331                   if (kind > 28)
332                      kind = 28;
333                   jjCheckNAdd(1);
334                   break;
335                default : break;
336             }
337          } while(i != startsAt);
338       }
339       else
340       {
341          int i2 = (curChar & 0xff) >> 6;
342          long l2 = 1L << (curChar & 077);
343          do
344          {
345             switch(jjstateSet[--i])
346             {
347                default : break;
348             }
349          } while(i != startsAt);
350       }
351       if (kind != 0x7fffffff)
352       {
353          jjmatchedKind = kind;
354          jjmatchedPos = curPos;
355          kind = 0x7fffffff;
356       }
357       ++curPos;
358       if ((i = jjnewStateCnt) == (startsAt = 3 - (jjnewStateCnt = startsAt)))
359          return curPos;
360       try { curChar = input_stream.readChar(); }
361       catch(java.io.IOException e) { return curPos; }
362    }
363 }
364 static final int[] jjnextStates = {
365 };
366
367 /** Token literal values. */
368 public static final String[] jjstrLiteralImages = {
369 "", null, null, null, null, "\164\162\165\145", "\146\141\154\163\145",
370 "\145\154\163\145", "\164\150\145\156", "\151\146", "\144\157", "\167\150\151\154\145",
    "\74\75",
371 "\76\75", "\74", "\76", "\75\75", "\72\75", "\41", "\156\145\167", "\54", "\73",
372 "\146\165\156", "\75\76", "\154\145\164", "\151\156", "\145\156\144", "\75", null, null,
373 "\53", "\55", "\52", "\57", "\50", "\51", "\73\73", };
374
375 /** Lexer state names. */
```

```
376 public static final String[] lexStateNames = {
377    "DEFAULT",
378 };
379 static final long[] jjtoToken = {
380    0x1fffffffe1L,
381 };
382 static final long[] jjtoSkip = {
383    0x1eL,
384 };
385 static protected SimpleCharStream input_stream;
386 static private final int[] jjrounds = new int[3];
387 static private final int[] jjstateSet = new int[6];
388 static protected char curChar;
389 /** Constructor. */
390 public ParserTokenManager(SimpleCharStream stream){
391    if (input_stream != null)
392       throw new TokenMgrError("ERROR: Second call to constructor of static lexer. You must
    use ReInit() to initialize the static variables.", TokenMgrError.STATIC_LEXER_ERROR);
393    input_stream = stream;
394 }
395
396 /** Constructor. */
397 public ParserTokenManager(SimpleCharStream stream, int lexState){
398    this(stream);
399    SwitchTo(lexState);
400 }
401
402 /** Reinitialise parser. */
403 static public void ReInit(SimpleCharStream stream)
404 {
405    jjmatchedPos = jjnewStateCnt = 0;
406    curLexState = defaultLexState;
407    input_stream = stream;
408    ReInitRounds();
409 }
410 static private void ReInitRounds()
411 {
412    int i;
413    jjround = 0x80000001;
414    for (i = 3; i-- > 0;)
415       jjrounds[i] = 0x80000000;
416 }
417
418 /** Reinitialise parser. */
419 static public void ReInit(SimpleCharStream stream, int lexState)
420 {
421    ReInit(stream);
422    SwitchTo(lexState);
423 }
424
425 /** Switch to specified lex state. */
426 static public void SwitchTo(int lexState)
427 {
428    if (lexState >= 1 || lexState < 0)
429       throw new TokenMgrError("Error: Ignoring invalid lexical state : " + lexState + ".
    State unchanged.", TokenMgrError.INVALID_LEXICAL_STATE);
430    else
431       curLexState = lexState;
432 }
433
434 static protected Token jjFillToken()
435 {
```

```
436    final Token t;
437    final String curTokenImage;
438    final int beginLine;
439    final int endLine;
440    final int beginColumn;
441    final int endColumn;
442    String im = jjstrLiteralImages[jjmatchedKind];
443    curTokenImage = (im == null) ? input_stream.GetImage() : im;
444    beginLine = input_stream.getBeginLine();
445    beginColumn = input_stream.getBeginColumn();
446    endLine = input_stream.getEndLine();
447    endColumn = input_stream.getEndColumn();
448    t = Token.newToken(jjmatchedKind, curTokenImage);
449
450    t.beginLine = beginLine;
451    t.endLine = endLine;
452    t.beginColumn = beginColumn;
453    t.endColumn = endColumn;
454
455    return t;
456 }
457
458 static int curLexState = 0;
459 static int defaultLexState = 0;
460 static int jjnewStateCnt;
461 static int jjround;
462 static int jjmatchedPos;
463 static int jjmatchedKind;
464
465 /** Get the next Token. */
466 public static Token getNextToken()
467 {
468   Token matchedToken;
469   int curPos = 0;
470
471   EOFLoop :
472   for (;;)
473   {
474    try
475    {
476       curChar = input_stream.BeginToken();
477    }
478    catch(java.io.IOException e)
479    {
480       jjmatchedKind = 0;
481       matchedToken = jjFillToken();
482       return matchedToken;
483    }
484
485    try { input_stream.backup(0);
486       while (curChar <= 32 && (0x100002600L & (1L << curChar)) != 0L)
487          curChar = input_stream.BeginToken();
488    }
489    catch (java.io.IOException e1) { continue EOFLoop; }
490    jjmatchedKind = 0x7fffffff;
491    jjmatchedPos = 0;
492    curPos = jjMoveStringLiteralDfa0_0();
493    if (jjmatchedKind != 0x7fffffff)
494    {
495       if (jjmatchedPos + 1 < curPos)
496          input_stream.backup(curPos - jjmatchedPos - 1);
497       if ((jjtoToken[jjmatchedKind >> 6] & (1L << (jjmatchedKind & 077))) != 0L)
```

```
498       {
499          matchedToken = jjFillToken();
500          return matchedToken;
501       }
502       else
503       {
504          continue EOFLoop;
505       }
506    }
507    int error_line = input_stream.getEndLine();
508    int error_column = input_stream.getEndColumn();
509    String error_after = null;
510    boolean EOFSeen = false;
511    try { input_stream.readChar(); input_stream.backup(1); }
512    catch (java.io.IOException e1) {
513       EOFSeen = true;
514       error_after = curPos <= 1 ? "" : input_stream.GetImage();
515       if (curChar == '\n' || curChar == '\r') {
516          error_line++;
517          error_column = 0;
518       }
519       else
520          error_column++;
521    }
522    if (!EOFSeen) {
523       input_stream.backup(1);
524       error_after = curPos <= 1 ? "" : input_stream.GetImage();
525    }
526    throw new TokenMgrError(EOFSeen, curLexState, error_line, error_column, error_after,
   curChar, TokenMgrError.LEXICAL_ERROR);
527  }
528 }
529
530 static private void jjCheckNAdd(int state)
531 {
532    if (jjrounds[state] != jjround)
533    {
534       jjstateSet[jjnewStateCnt++] = state;
535       jjrounds[state] = jjround;
536    }
537 }
538 static private void jjAddStates(int start, int end)
539 {
540    do {
541       jjstateSet[jjnewStateCnt++] = jjnextStates[start];
542    } while (start++ != end);
543 }
544 static private void jjCheckNAddTwoStates(int state1, int state2)
545 {
546    jjCheckNAdd(state1);
547    jjCheckNAdd(state2);
548 }
549
550 }
551
```