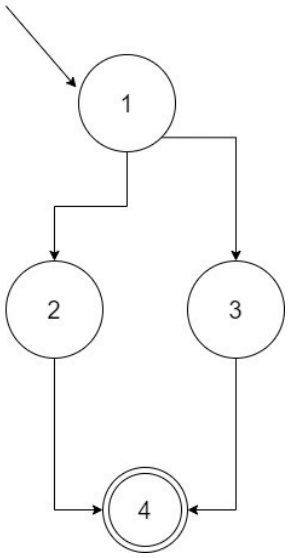
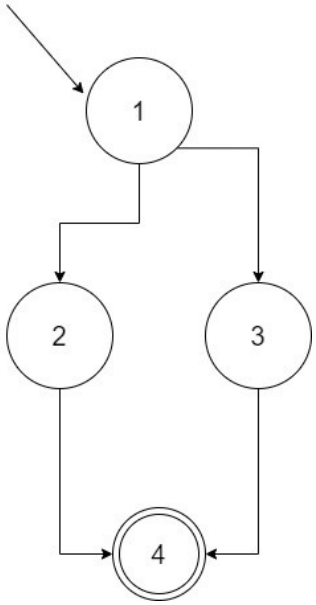
OVERVIEW:

Here are the following CFG(s) for the following methods. Get_char(), unget_char(), and unget_error() do not have branches therefore as per requirement they will be skipped. Note the code in the catch clauses are also omitted as per requirement of this project assignment. Note that the CFG, tables and code snippets are followed in sequential order and the code snippet are referenced from the block tables. Defs() and uses() labels for the CFG(s) are also excluded from it since it never ask explicitly as a requirement in the project assignment and also causes significant clutter in the CFG diagram. Note that the CFG of the End-to-End system/System-Level are found at the last page of this document. Turn to the next page to start the review.



Method 1			
Block	Lines	Entry	Exit
1	35, 36	35	36
2	38	38	38
3	45, 46	45	46
4	57	57	57

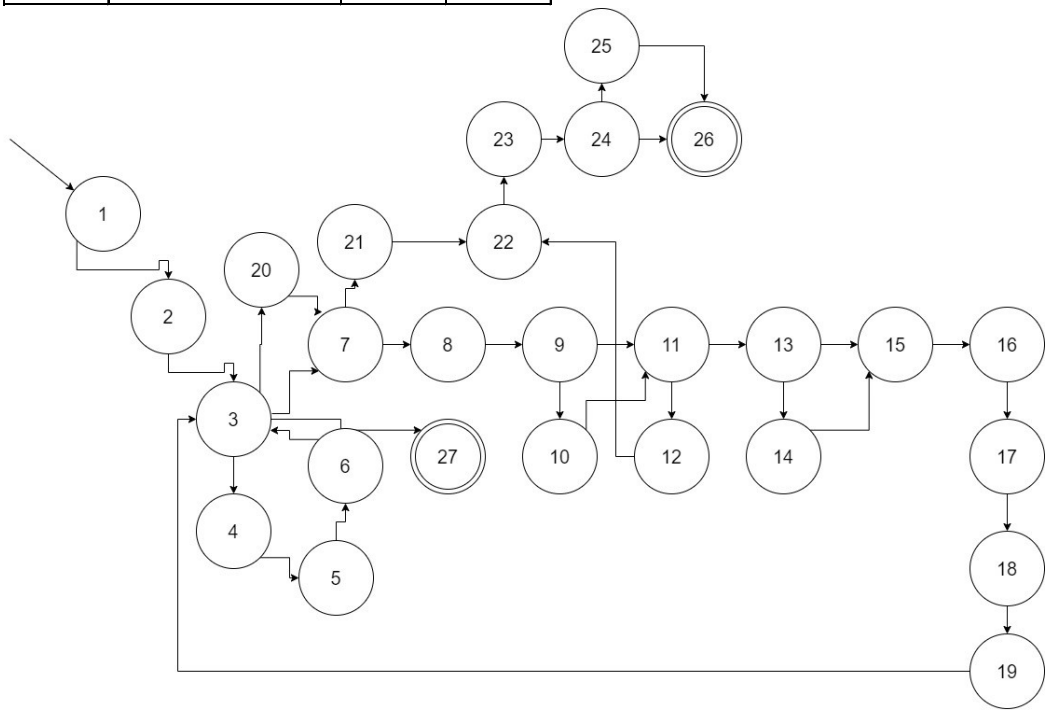
```
25          /************************************************/
26          /* NMAE:        open_character_stream           */
27          /* INPUT:       a filename                      */
28          /* OUTPUT:      a BufferedReader */
29          /* DESCRIPTION: when not given a filename,      */
30          /*              open stdin,otherwise open       */
31          /*              the existed file                */
32          /************************************************/
33          BufferedReader open_character_stream(String fname)
34          {
35              BufferedReader br = null;
36              if (fname == null)
37              {
38                  br = new BufferedReader(new InputStreamReader(System.in));
39                  //EDIT: INSERT UNUSED FUNCTION UNGET_ERROR HERE
40                  unget_error(br);
41              }
42
43              else
44              {
45                  try
46                  {
47                      FileReader fr = new FileReader(fname);
48                      br = new BufferedReader(fr);
49                  }
50
51                  catch (FileNotFoundException e)
52                  {
53                      System.out.print("The file " + fname +" doesn't exists\n");
                        e.printStackTrace();
55                  }
56              }
57
58              //EDIT: change null to br
59              return br;
60          }
61
```



Method 4			
Block	Lines	Entry	Exit
1	113, 116	113	116
2	118	118	118
3	123	123	123
4	126	126	126

```
106         /*********************************************************/
107         /* NAME:         open_token_stream                       */
108         /* INPUT:        a filename                              */
109         /* OUTPUT:       a BufferedReader              */
110         /* DESCRIPTION: when filename is EMPTY,choice standard  */
111         /*               input device as input source            */
112         /*********************************************************/
113         BufferedReader open_token_stream(String fname)
114         {
115             BufferedReader br;
116
117             //EDIT:change null to "" and add curly brackets
118             if(fname.equals(" "))
119             {
120                 br=open_character_stream(null);
121             }
122
123             else
124             {
125                 br=open_character_stream(fname);
126             }
127
128             return br;
129         }
130
```

Method 5			
Block	Lines	Entry	Exit
1	141, 142, 145, 147, 150	141	150
2	151	151	151
3	153	153	153
27	155	155	155
4	158	158	158
5	161	161	161
6	163, 164	163	164
7	172, 174	172	174
8	177	177	177
9	180	180	180
10	182	182	182
11	185	185	185
12	187	187	187
13	190, 191	190	191
14	193, 194	193	194
15	196	196	196
16	199	199	199
17	201, 202, 203, 204	201	204
18	206	206	206
19	209	209	209
20	214, 215	214	215
21	221, 222	221	222
22	225	225	225
23	227, 228	227	228
24	231	231	231
25	233, 234	233	234
26	243	243	243



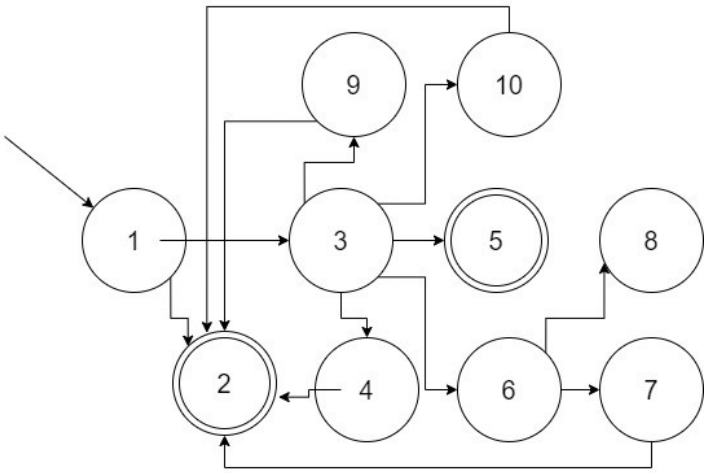
(continued to next two pages for source code snippet)

```
/*************************************************************/
/* NAME :       get_token                                    */
/* INPUT:       a BufferedReader          */
/* OUTPUT:      a token string                                */
/* DESCRIPTION: according the syntax of tokens,dealing  */
/*              with different case  and get one token  */
/*************************************************************/
String get_token(BufferedReader br)
{
   //EDIT: int i and j will be comment out
  //int i=0,j;

    int id=0;
    int res = 0;

    //EDIT: remove '\0' in ch variable
    char ch;

    StringBuilder sb = new StringBuilder();

    try
    {
        res = get_char(br);

        if (res == -1)
        {
            return null;
        }

        ch = (char)res;

        //EDIT:change condition from '\t' to ' '
        while(ch==' '||ch=='\n' || ch == '\r')      /* strip all blanks until meet characters */
        {
            res = get_char(br);
            ch = (char)res;
        }

        if(res == -1)
        {
            return null;
        }

        sb.append(ch);

        //EDIT: change conditions from is_spec_symbol(ch)==true to is_spec_symbol(ch)
```

```
        //EDIT: change conditions from is_spec_symbol(ch)==true to is_spec_symbol(ch)
        if(is_spec_symbol(ch))
        {
            return sb.toString();
        }

        if(ch =='"')
        {
            id=2; /* prepare for string */
        }

        if(ch ==59)
        {
            id=1; /* prepare for comment */
        }

        res = get_char(br);
        if (res == -1)
        {
            unget_char(ch,br);
            return sb.toString();
        }
        ch = (char)res;

        //EDIT: CHANGE is_token_end(id,res) == false TO !is_token_end(id, res)
        while (!is_token_end(id, res))/* until meet the end character */
        {
            sb.append(ch);
            br.mark(4);
            res = get_char(br);
            if (res == -1)
            {
                break;
            }

            ch = (char)res;
        }

        if(res == -1)       /* if end character is eof token    */
        {
            unget_char(ch,br);        /* then put back eof on token_stream */
            return sb.toString();
        }
```

```
220                 //EDIT: CHANGE is_spec_symbol(ch)==true TO is_spec_symbol(ch)
221                 if(is_spec_symbol(ch))      /* if end character is special_symbol */
222                 {
223                     unget_char(ch,br);         /* then put back this character       */
224                     return sb.toString();
225                 }
226
227                 if(id==1)                   /* if end character is " and is string */
228                 {
229                     sb.append(ch);
230                     return sb.toString();
231                 }
232
233                 if(id==0 && ch==59)         /* when not in string or comment,meet ";" */
234                 {
235                     unget_char(ch,br);        /* then put back this character         */
236                     return sb.toString();
237                 }
238             }
239
240             catch (IOException e)
241             {
242                 e.printStackTrace();
243             }
244
245             return sb.toString();                   /* return nomal case token
246         }
247
```

Method 6			
Block	Lines	Entry	Exit
1	254	254	254
2	256	256	256
3	259, 260	259	260
4	263	263	263
5	270	270	270
6	273	273	273
7	275	275	275
8	280	280	280
9	287	287	287
10	292	292	292

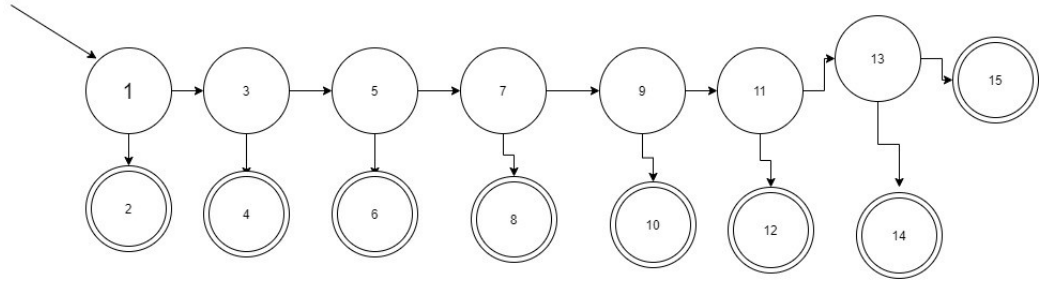



```
248         /*************************************************************/
249         /* NAME:         is_token_end                                */
250         /* INPUT:        a character,a token status                  */
251         /* OUTPUT:       a BOOLEAN value                             */
252         /*************************************************************/
253         static boolean is_token_end(int str_com_id, int res)
254         {
255             //EDIT: ADD CURLY BRACKET TO IF AND ELSE STATEMENTS
256             if(res==-1)
257             {
258                 return(true); /* is eof token? */
259             }
260
261             char ch = (char)res;
262             if(str_com_id==1)          /* is string token */
263             {
264                 //EDIT: INSERT || AND ADD CURLY BRACKETS
266                 if(ch=='"' || ch=='\n' || ch == '\r')
267                 {
268                     /* for string until meet another " */
269                     return true;
270                 }
271
272                 else
273                     return false;
274             }
275
276             if(str_com_id==2)    /* is comment token */
278             {
279                 if(ch=='\n' || ch == '\r' || ch=='\t')      /* for comment until meet end of line */
280                 {
281                     return true;
282                 }
283
284                else
285                 {
286                     return false;
287                 }
288             }
289
290             //EDIT: CHANGE is_spec_symbol(ch)==true TO is_spec_symbol(ch)
291             if(is_spec_symbol(ch))
292             {
293                 return true;
             } /* is special_symbol? */
```

(continue to next page for source code snippet)

```
293
            if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59)
295         {
296             return true;
297         }
298
299         return false;               /* other case,return FALSE */
300     }
301
```

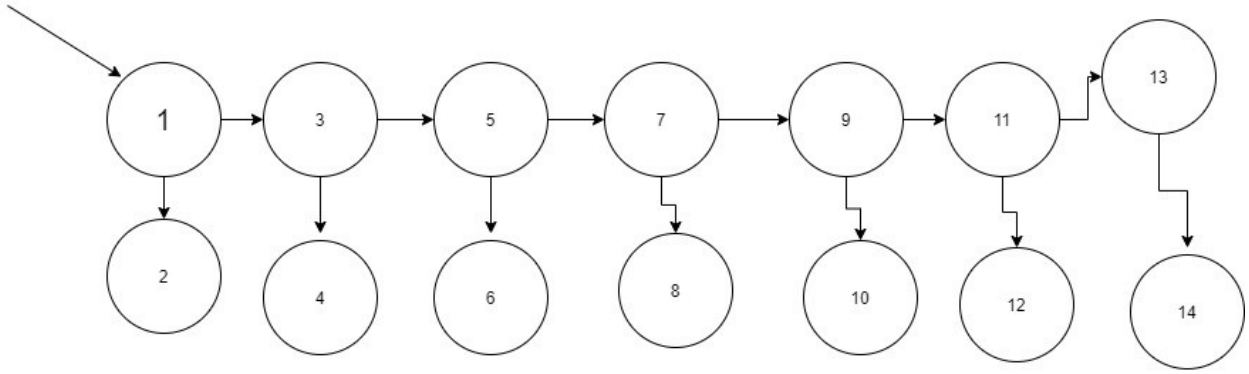
Method 7			
Block	Lines	Entry	Exit
1	310	310	310
2	312	312	312
3	315	315	315
4	317	317	317
5	320	320	320
6	322	322	322
7	325	325	325
8	327	327	327
9	330	330	330
10	332	332	332
11	335	335	335
12	337	337	337
13	340	340	340
14	342	342	342
15	345	345	345




```
302         /*******************************************************/
303         /* NAME :        token_type                            */
304         /* INPUT:        a token              */
305         /* OUTPUT:       an integer value                      */
306         /* DESCRIPTION: the integer value is corresponding     */
307         /*               to the different token type           */
308         /*******************************************************/
309         static int token_type(String tok)
310         {
311             //Edit: add curly brackets
312             if(is_keyword(tok))
313             {
314                return(keyword);
315             }
316
317             if(is_spec_symbol(tok.charAt(0)))
318             {
319                return(spec_symbol);
320             }
321
322             if(is_identifier(tok))
323             {
324                return(identifier);
325             }
326
327             if(is_num_constant(tok))
328             {
329                return(num_constant);
330             }
```

(continue to next page for source code snippet)

```
332             if(is_str_constant(tok))
333             {
334                 return(str_constant);
335             }
336
337             if(is_char_constant(tok))
338             {
339                 return(char_constant);
340             }
341
342             if(is_comment(tok))
343             {
344                 return(comment);
345             }
346
347             return(error);                  /* else look as error token */
348         }
349
```

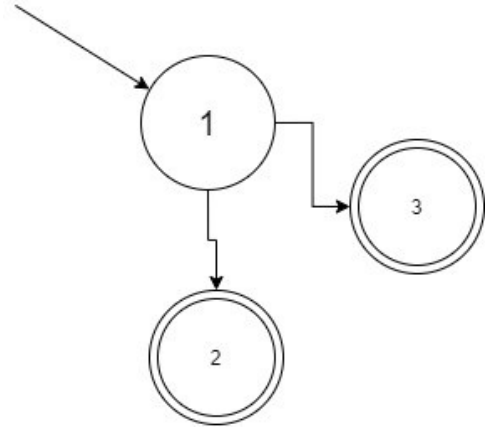


Method 8			
Block	Lines	Entry	Exit
1	354, 355,357	354	357
2	359	359	359
3	362	362	362
4	364	364	364
5	367	367	367
6	369	369	369
7	372	372	372
8	374	374	374
9	377	377	377
10	379	379	379
11	384	384	384
12	386	386	386
13	389	389	389
14	391	391	391

(continue to next page for source code snippet)

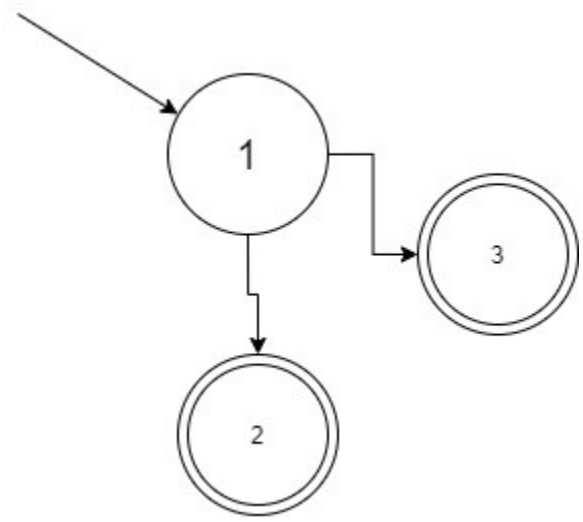
```
350         /*****************************************************/
351         /* NAME:        print_token                          */
352         /* INPUT:       a token                              */
353         /*****************************************************/
354         void print_token(String tok)
355         {
356             int type;
357             type=token_type(tok);
358
359             if(type==error)
360             {
361                 System.out.print("error,\"" + tok + "\".\n");
362             }
363
364             if(type==keyword)
365             {
366                 System.out.print("keyword,\"" + tok + "\".\n");
367             }
368
369             if(type==spec_symbol)
370             {
371                 print_spec_symbol(tok);
372             }
373
374             if(type==identifier)
375             {
376                 System.out.print("identifier,\"" + tok + "\".\n");
377             }
378
379             if(type==num_constant)
380             {
381                 System.out.print("numeric," + tok + ".\n");
382             }
383
384             //EDIT ADD IF STATEMENT WITH CONDITION FOR STRING CONSTANT
385
386             if(type==str_constant)
387             {
388                 System.out.print("string," + tok + ".\n");
389             }
390
391             if(type==char_constant)
392             {
393                 System.out.print("character,\"" + tok.charAt(1) + "\".\n");
394             }
395         }
```

Method 9			
Block	Lines	Entry	Exit
1	405	405	405
2	407	407	407
3	412	412	412




```
399
400         /*************************************/
401         /* NAME:         is_comment          */
402         /* INPUT:        a token */
403         /* OUTPUT:       a BOOLEAN value     */
404         /*************************************/
405         static boolean is_comment(String ident)
406         {
            if( ident.charAt(0) == 59)    /* the char is 59    */
408         {
409             return true;
410         }
411
412         else
413         {
414             return false;
415         }
416         }
417
```

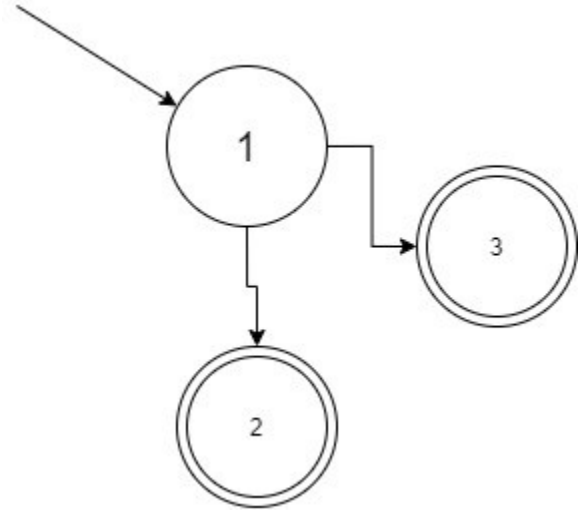
Method 10			
Block	Lines	Entry	Exit
1	423	423	423
2	425	425	425
3	430	430	430




```
418         /*************************************/
419         /* NAME:         is_keyword          */
420         /* INPUT:        a token */
421         /* OUTPUT:       a BOOLEAN value     */
422         /*************************************/
423         static boolean is_keyword(String str)
424         {
426             if (str.equals("and") || str.equals("or") || str.equals("if") || str.equals("xor")||str.equals("lambda")||str.equals("=>"))
427             {
428                 return true;
429             }
430
431             else
432             {
433                 return false;
434             }
435         }
```

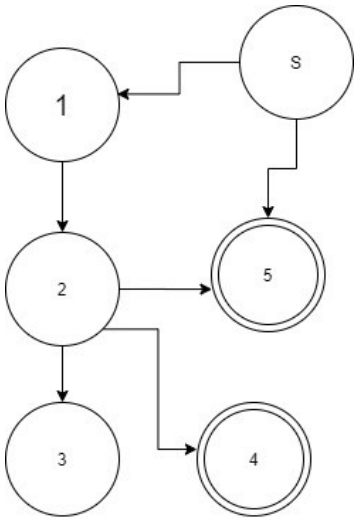
Method 11			
Block	Lines	Entry	Exit
1	442	442	442
2	444	444	444
3	448	448	448

```
436         /*************************************/
437         /* NAME:         is_char_constant     */
438         /* INPUT:        a token */
439         /* OUTPUT:       a BOOLEAN value      */
440         /*************************************/
441         static boolean is_char_constant(String str)
442         {
443             //Edit: change > to >=
            if (str.length() >= 2 && str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
445             {
446                 return true;
447             }
448             else
449             {
450                 return false;
451             }
452         }
453
```



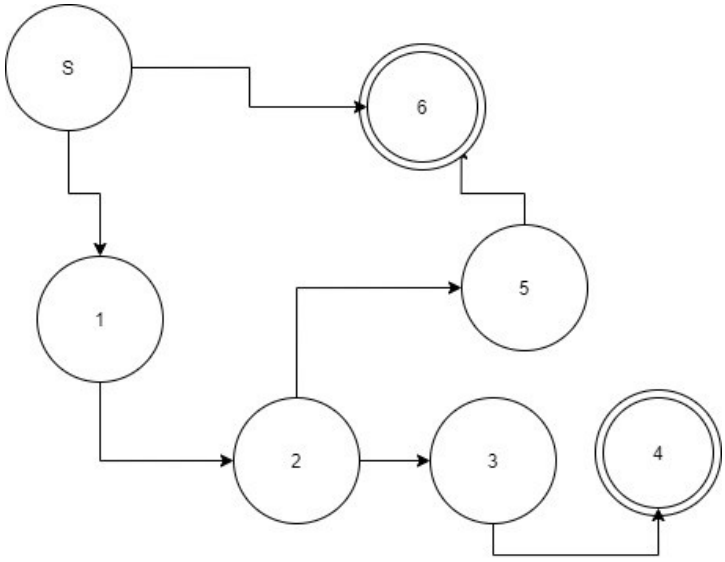
Method 12			
Block	Lines	Entry	Exit
S	457	457	457
1	459, 461	459	461
2	464	464	464
3	467	467	467
4	471	471	471
5	477	477	477
6	482	482	482

```
454         /*************************************/
455         /* NAME:         is_num_constant     */
456         /* INPUT:        a token */
457         /* OUTPUT:       a BOOLEAN value     */
458         /*************************************/
459         static boolean is_num_constant(String str)
460         {
461           int i=1;
462
463           if(Character.isDigit(str.charAt(0)))
464             {
465                 //Edit: change <= to <
466                 while (i < str.length() && str.charAt(i) != '\0')   /* until meet token end sign */
467                 {
468                     //Edit: change i+1 to i
469                     if(Character.isDigit(str.charAt(i)))
470                     {
471                         i++;
472                     }
473                     else
474                     {
475                         return false;
476                     }
477                 }                           /* end WHILE */
478
479                 return true;
480             }
481
482           else
483           {
484             return false;
485           } /* other return FALSE */
486
487         }
488
```

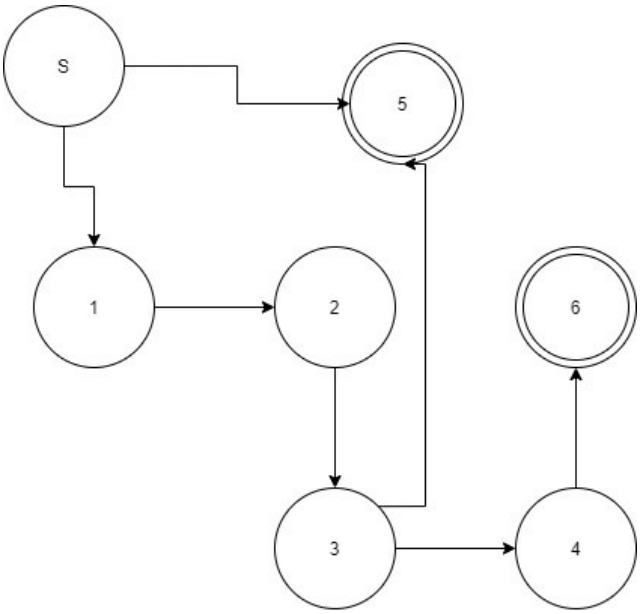


Method 13			
Block	Lines	Entry	Exit
1	494, 495	494	495
2	497	497	497
3	499	499	499
4	501	501	501
5	506	506	506
6	511	511	511

```
489         /*************************************/
490         /* NAME:         is_str_constant      */
491         /* INPUT:        a token */
492         /* OUTPUT:       a BOOLEAN value      */
493         /*************************************/
494         static boolean is_str_constant(String str)
495         {
496             int i=1;
497             if(str.charAt(0) =='"')
498             {
499                 while (i < str.length() && str.charAt(0)!='\0')  /* until meet the token end sign */
500                 {
501                     if(str.charAt(i)=='"')
502                     {
503                         return true;
504                     }
505                         /* meet the second '"'           */
506                     else
507                     {
508                         i++;
509                     }
510                 }               /* end WHILE */
511
512                 //EDIT: change return true to false
513                 return false;
514             }
515
516             else
517             {
518                 return false;
519             }
520             /* other return FALSE */
521         }
522
```

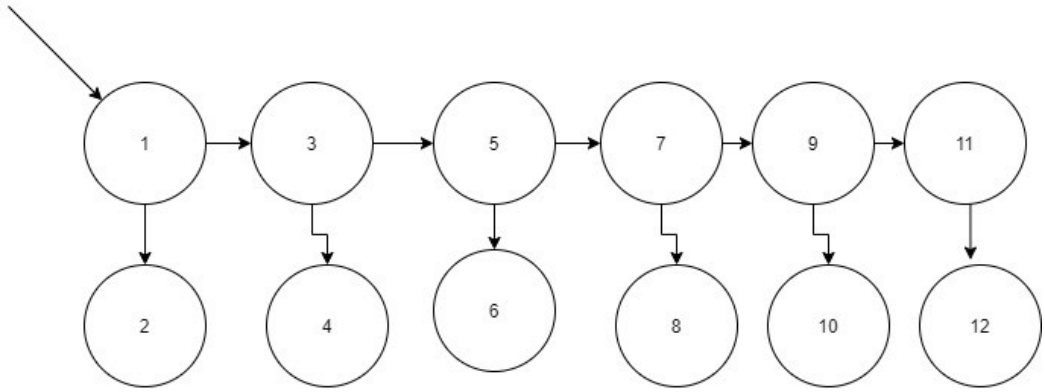


Method 14			
Block	Lines	Entry	Exit
1	528, 530	528	530
2	532	532	532
3	534	534	534
4	536	536	536
5	540	540	540
6	544	544	544



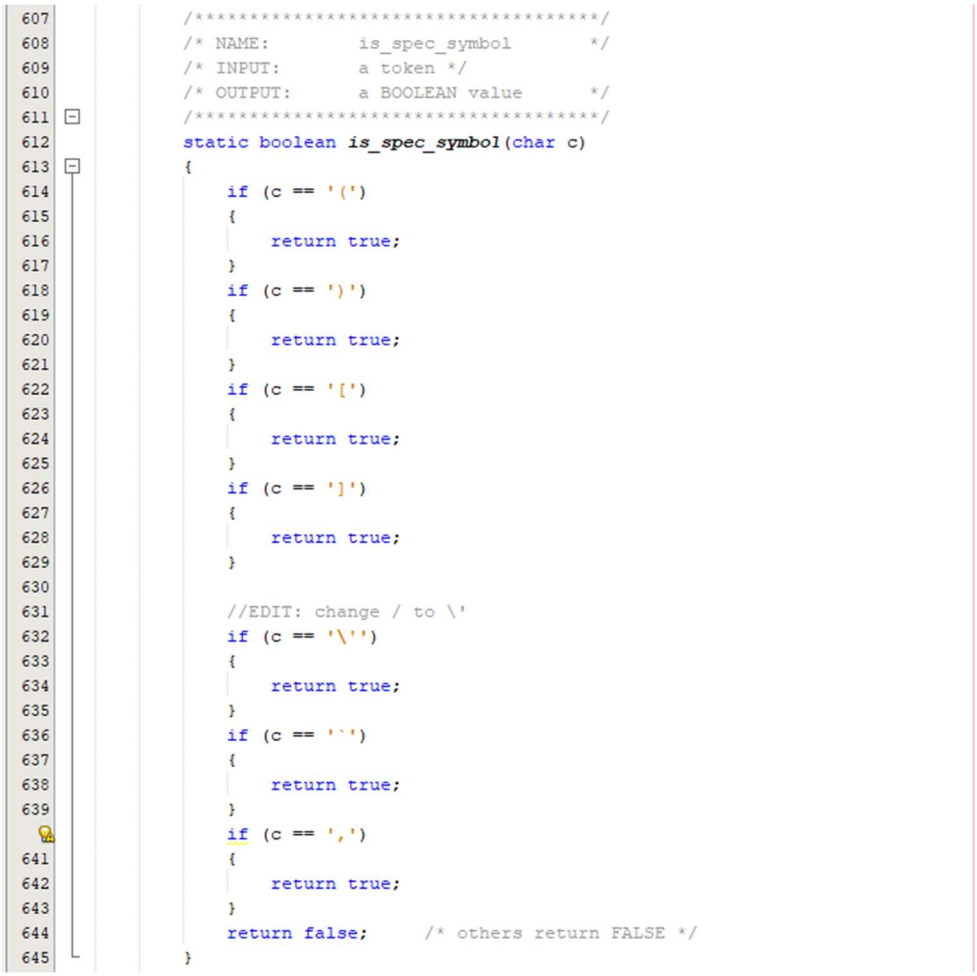

```
523         /*************************************/
524         /* NAME:         is_identifier         */
525         /* INPUT:        a token */
526         /* OUTPUT:       a BOOLEAN value       */
527         /*************************************/
528         static boolean is_identifier(String str)
529         {
530           int i=0;
531           //Edit: change condition value 0 to i
532             if(Character.isLetter(str.charAt(i)))
533             {
534                 while(i < str.length() && str.charAt(i) !='\0')   /* unti meet the end token sign */
535                 {
536                     if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
537                     {
538                        i++;
539                     }
540                     else
541                     {
542                        return false;
543                     }
544                 }       /* end WHILE */
545                 //EDIT: change false to true
546                 return true;
547             }
548
549             else
550             {
551                 //EDIT: CHANGE TRUE TO FALSE
552                 return false;
553             }
554         }
555
```

Method 16			
Block	Lines	Entry	Exit
1	574	574	574
2	576	576	576
3	579	579	579
4	581	581	581
5	584	584	584
6	586	586	586
7	589	589	589
8	591	591	591
9	594	594	594
10	596	596	596
11	599	599	599
12	601	601	601



(continue to next page for source code snippet)

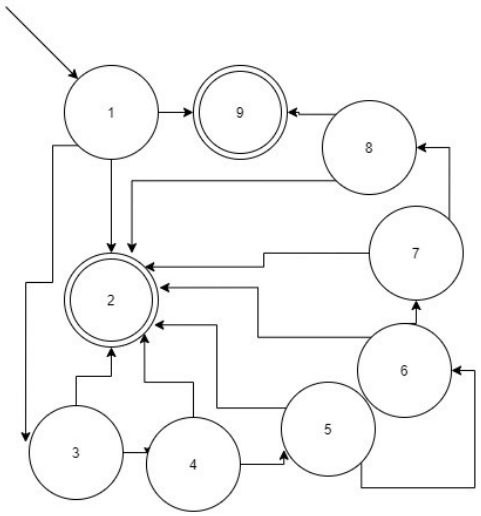
```
566         /*************************************************/
567         /* NAME:         print_spec_symbol               */
568         /* INPUT:        a spec_symbol token */
569         /* OUTPUT :      print out the spec_symbol token  */
570         /*               according to the form required   */
571         /*************************************************/
572         static void print_spec_symbol(String str)
573         {
574             //Edit remove all returns in the if statements
575             //Edit: change { to (
576             if(str.equals("("))
577             {
578                 System.out.print("lparen.\n");
579             }
580 
581             if (str.equals(")"))
582             {
583                 System.out.print("rparen.\n");
584             }
585 
586             if (str.equals("["))
587             {
588                 System.out.print("lsquare.\n");
589             }
590 
591             if (str.equals("]"))
592             {
593                 System.out.print("rsquare.\n");
594             }
595 
596             if (str.equals("'"))
597             {
598                 System.out.print("quote.\n");
599             }
600 
601             if (str.equals("`"))
602             {
603                 System.out.print("bquote.\n");
604             }
605         }
606 
```

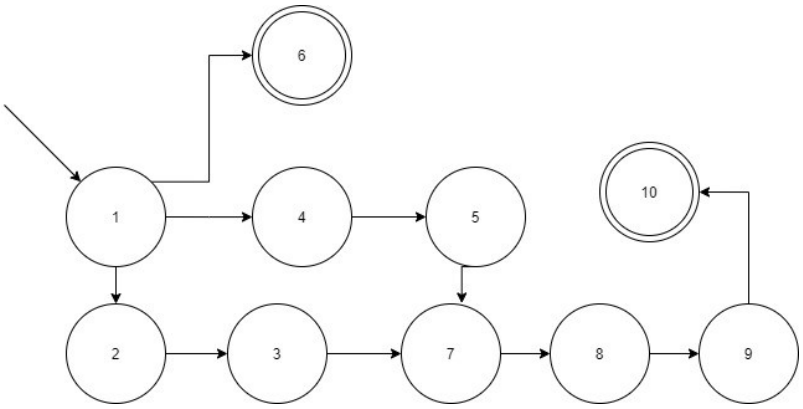


```
607         /*************************************/
608         /* NAME:         is_spec_symbol      */
609         /* INPUT:        a token */
610         /* OUTPUT:       a BOOLEAN value     */
611         /*************************************/
612         static boolean is_spec_symbol(char c)
613         {
614             if (c == '(')
615             {
616                 return true;
617             }
618             if (c == ')')
619             {
620                 return true;
621             }
622             if (c == '[')
623             {
624                 return true;
625             }
626             if (c == ']')
627             {
628                 return true;
629             }
630 
631             //EDIT: change / to \'
632             if (c == '\'')
633             {
634                 return true;
635             }
636             if (c == '`')
637             {
638                 return true;
639             }
641             if (c == ',')
641             {
642                 return true;
643             }
644             return false;     /* others return FALSE */
645         }
```

Method 17			
Block	Lines	Entry	Exit
1	612	612	612
2	614	614	614
3	616	616	616
4	620	620	620
5	624	624	624
6	630	630	630
7	634	634	634
8	638	638	638
9	642	642	642

(continue to next page for CFG diagram snippet)


1
9
8
7
2
6
5
3
4




```
647         public static void main(String[]args) throws IOException
648         {
            String fname = null;
            if (args.length == 0)
651         {   /* if not given filename,take as '""' */
652             //EDIT: change new string() to " "
653             fname = " ";
654         }
655
656         else if(args.length == 1)
657         {
658             //EDIT: change value 1 to 0 in bracket
659             fname = args[0];
660         }
661
662         else
663         {
664             System.out.print("Error!,please give the token stream\n");
665             System.exit(0);
666         }
667
668         Printtokens t = new Printtokens();
669         BufferedReader br = t.open_token_stream(fname);     /* open token stream */
670         String tok = t.get_token(br);
671         while (tok != null)
672         {   /* take one token each time until eof */
673             t.print_token(tok);
674             tok = t.get_token(br);
675         }
676
677         System.exit(0);
678     }
679 }
```

Method 18 : Main			
Block	Lines	Entry	Exit
1	647, 648	647	648
2	651	651	651
3	654	654	654
4	657	657	657
5	662	662	662
6	663, 664, 665	663	665
7	668, 669, 670	668	670
8	671	671	671
9	673, 674	673	674
10	677	677	677

End-To-End System-Level block			
Block	Lines	Entry	Exit
1	14,15,16,17,18,19,20,21	14	21

End-To-End System-Level Methods	
Method	Method edge coverage
1	
2	
3	
4	1
5	2, 17, 3,6
6	17
7	9, 10, 11, 12, 13, 14, 17
8	7, 16
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	4, 5, 8