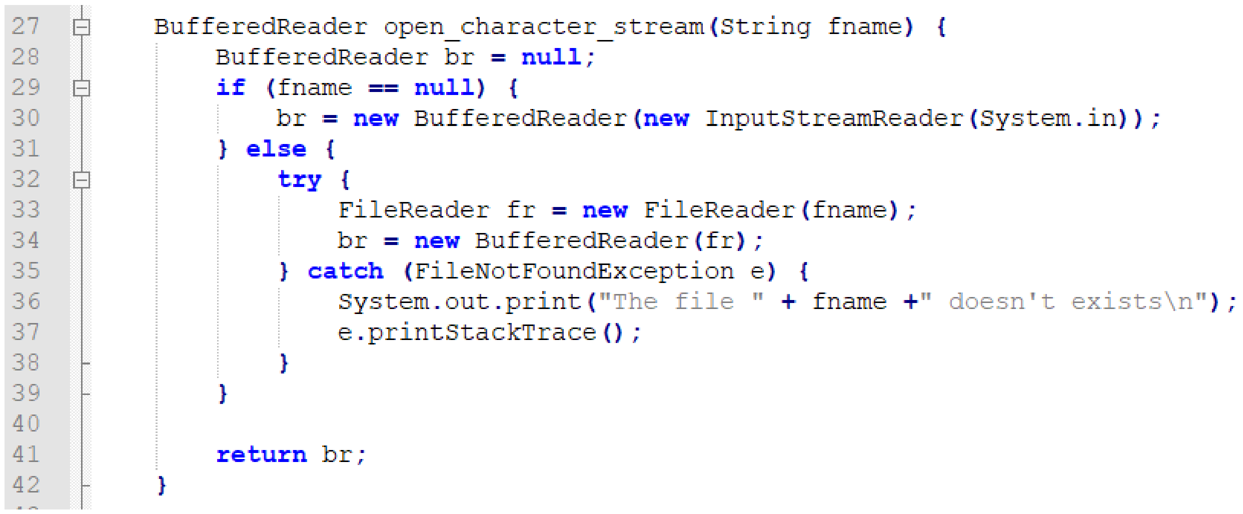
CSE 5321 Software Testing - Project – Summer 2018

CONTROL FLOW GRAPH

1. open_character_stream(String fname)

Screenshot of Function:

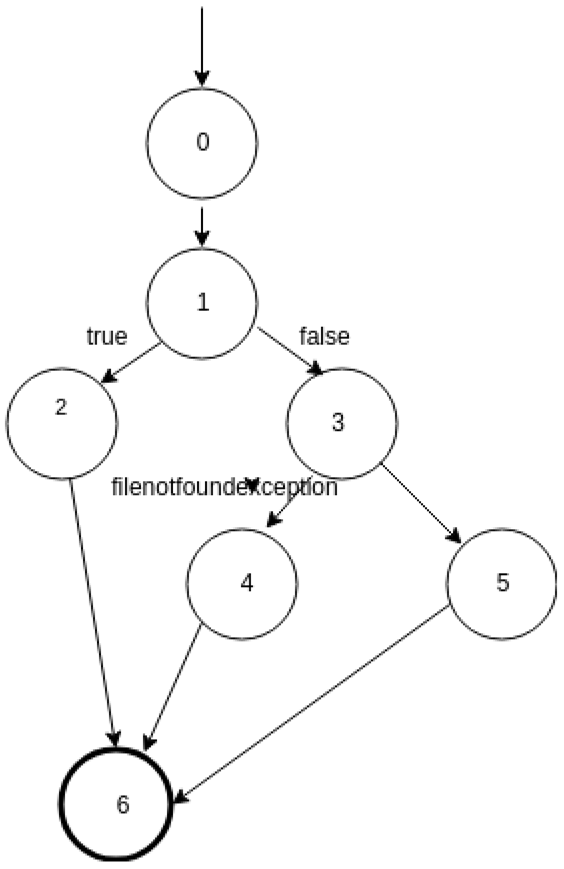


```
27   BufferedReader open_character_stream(String fname) {
28       BufferedReader br = null;
29       if (fname == null) {
30           br = new BufferedReader(new InputStreamReader(System.in));
31       } else {
32           try {
33               FileReader fr = new FileReader(fname);
34               br = new BufferedReader(fr);
35           } catch (FileNotFoundException e) {
36               System.out.print("The file " + fname +" doesn't exists\n");
37               e.printStackTrace();
38           }
39       }
40
41       return br;
42   }
```

Block table for the Function:

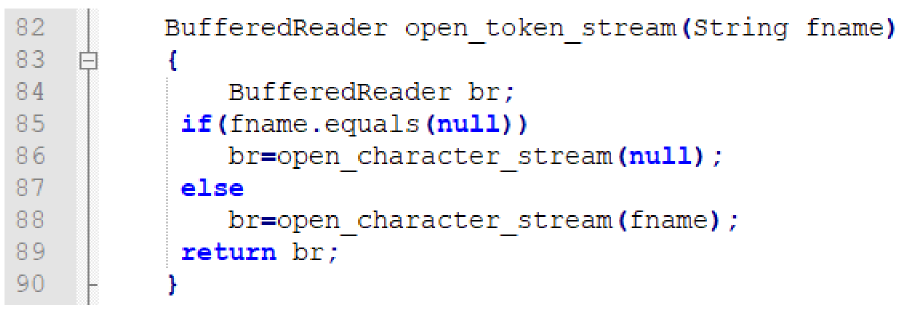
BLOCK	LINES	ENTRY	EXIT
0	27	27	27
1	28-29	28	29
2	30	30	30
3	31-32	31	32
4	35-38	35	38
5	33-34	33	34
6	41	41	41

Control Flow Graph for the Function:



2. open_token_stream(String fname)

Screenshot of Function:

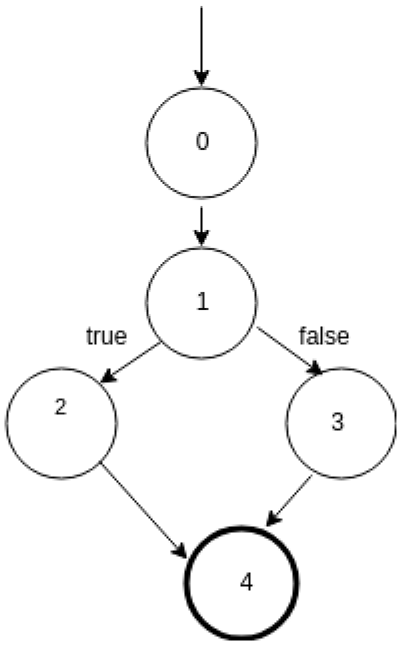


```
BufferedReader open_token_stream(String fname)
{
    BufferedReader br;
 if(fname.equals(null))
    br=open_character_stream(null);
 else
    br=open_character_stream(fname);
 return br;
}
```

Block table for the Function:

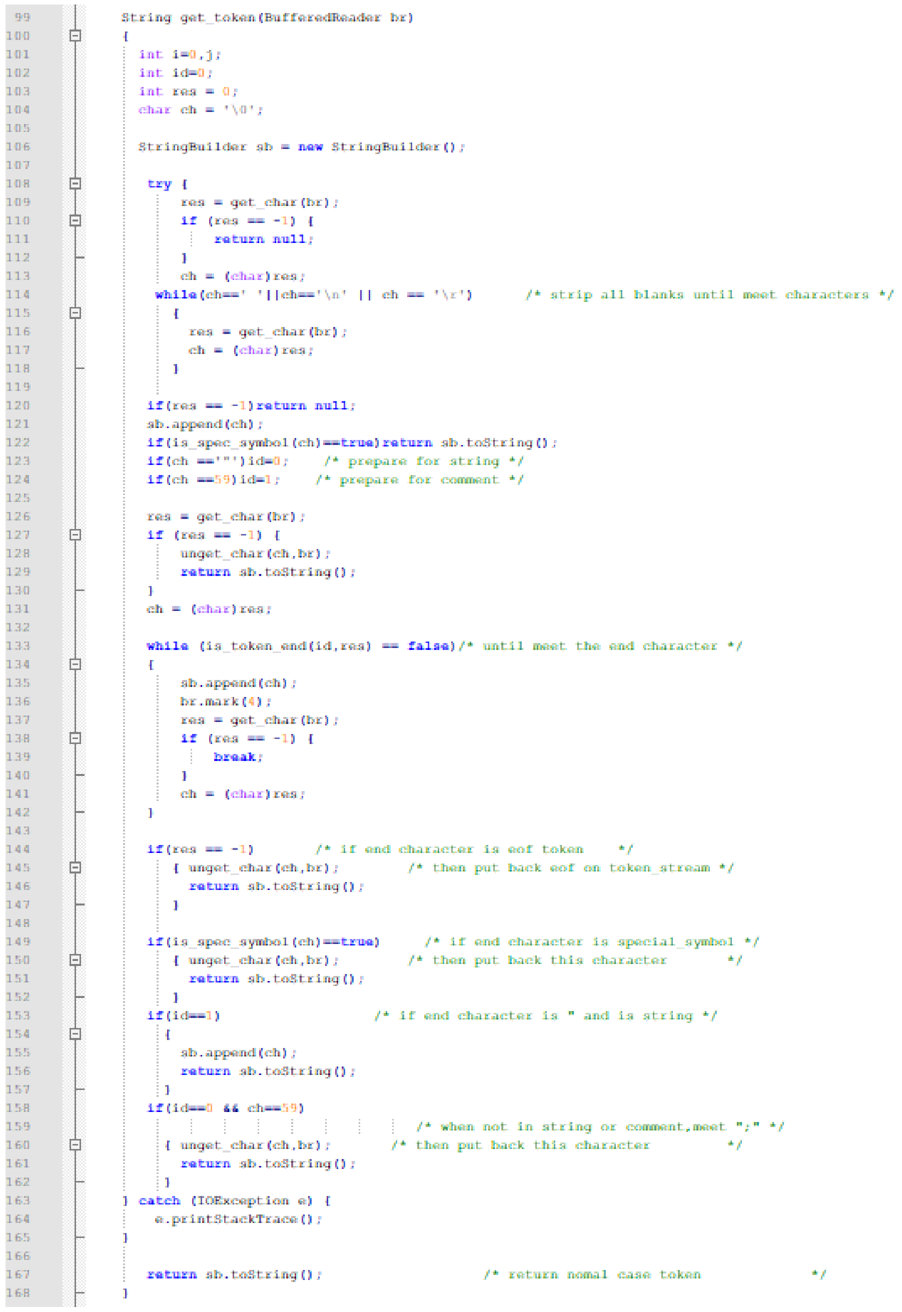
BLOCK	LINES	ENTRY	EXIT
0	82	82	82
1	84-85	84	85
2	86	86	86
3	88	88	88
4	89	89	89

Control Flow Graph for the Function:



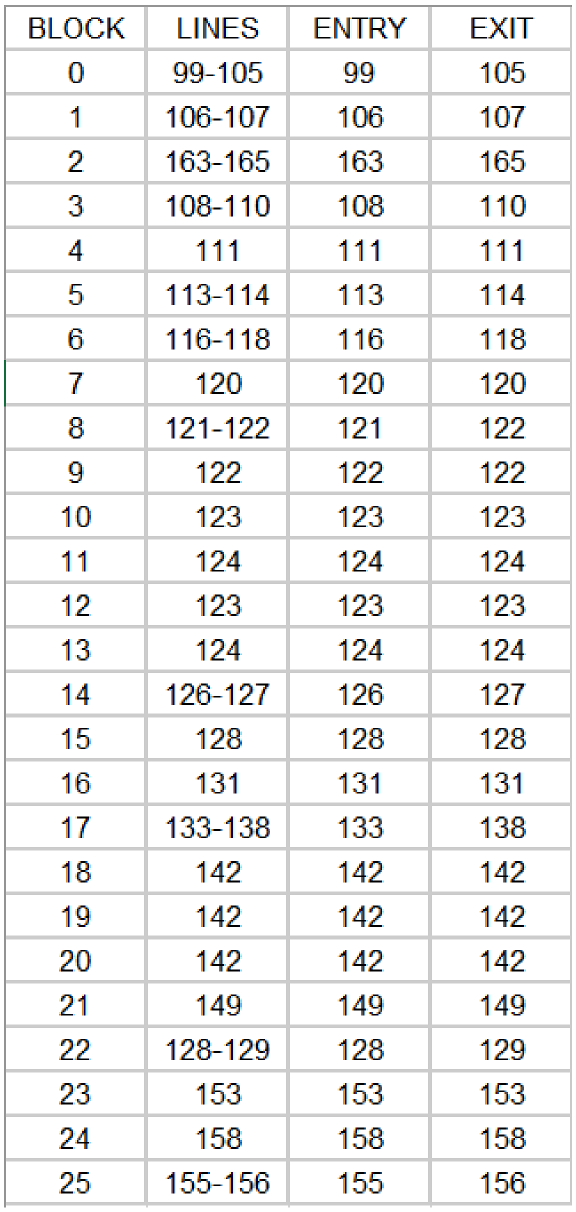
3. String get_token(BufferedReader br)

Screenshot of Function:



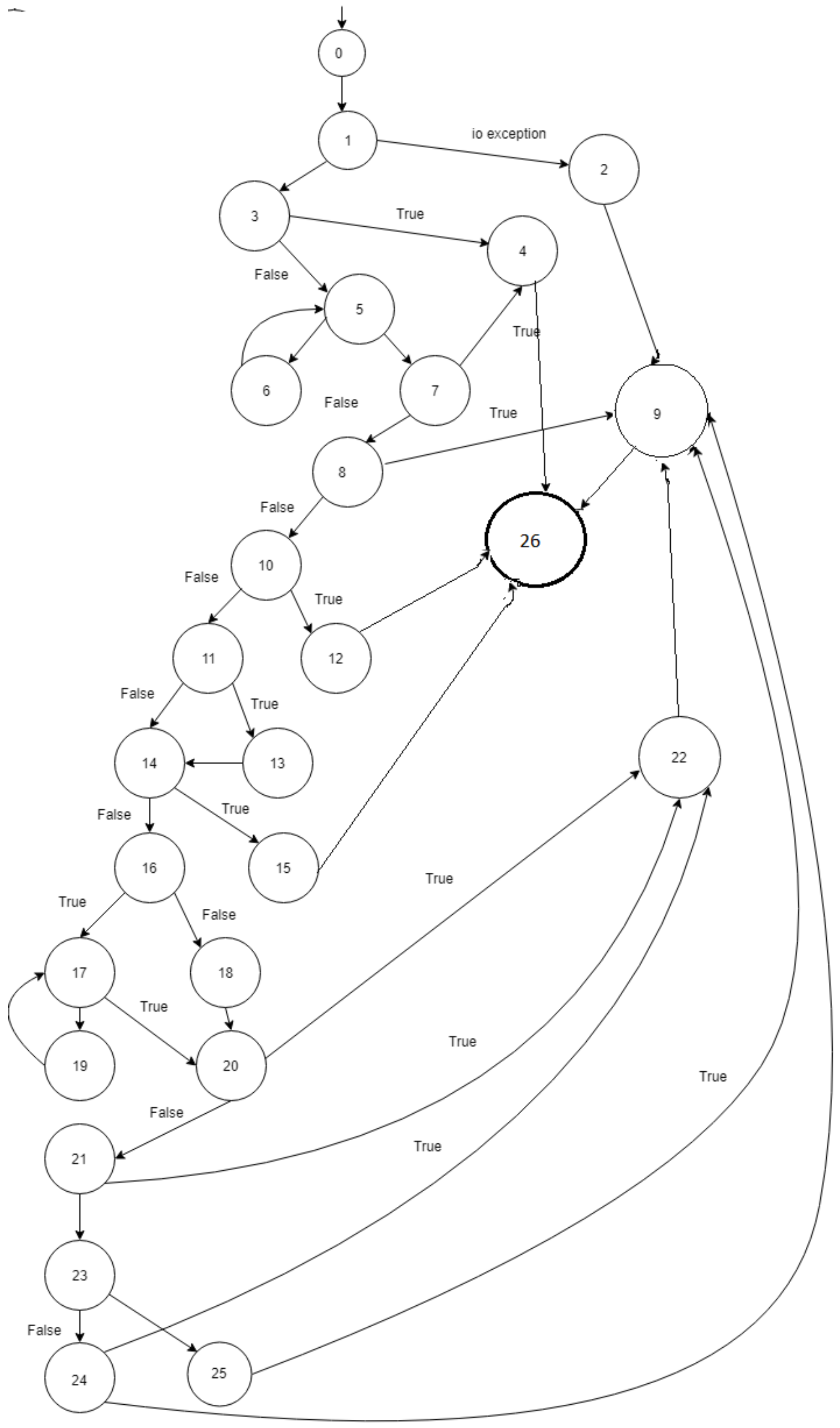
```
String get_token(BufferedReader br)
{
  int i=0,j;
  int id=0;
  int res = 0;
  char ch = '\0';

  StringBuilder sb = new StringBuilder();

   try {
      res = get_char(br);
      if (res == -1) {
          return null;
      }
      ch = (char)res;
    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);
   if(is_spec_symbol(ch)==true)return sb.toString();
   if(ch =='"')id=0;     /* 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;

   while (is_token_end(id,res) == false)/* 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();
     }

   if(is_spec_symbol(ch)==true)      /* if end character is special_symbol */
     { unget_char(ch,br);          /* then put back this character       */
       return sb.toString();
     }
   if(id==1)                    /* if end character is " and is string */
    {
      sb.append(ch);
      return sb.toString();
    }
   if(id==0 && ch==59)
                                /* when not in string or comment,meet ";" */
    { unget_char(ch,br);        /* then put back this character          */
      return sb.toString();
    }
 } catch (IOException e) {
    e.printStackTrace();
 }

   return sb.toString();                    /* return nomal case token             */
 }
```

Block table for the Function:



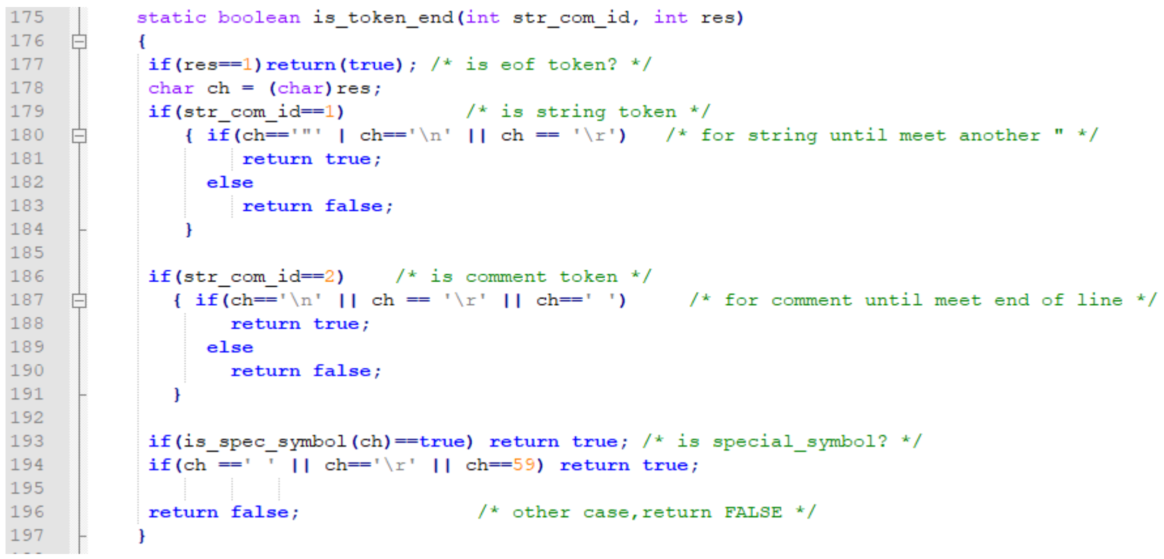
BLOCK	LINES	ENTRY	EXIT
0	99-105	99	105
1	106-107	106	107
2	163-165	163	165
3	108-110	108	110
4	111	111	111
5	113-114	113	114
6	116-118	116	118
7	120	120	120
8	121-122	121	122
9	122	122	122
10	123	123	123
11	124	124	124
12	123	123	123
13	124	124	124
14	126-127	126	127
15	128	128	128
16	131	131	131
17	133-138	133	138
18	142	142	142
19	142	142	142
20	142	142	142
21	149	149	149
22	128-129	128	129
23	153	153	153
24	158	158	158
25	155-156	155	156

Control Flow Graph for the Function:



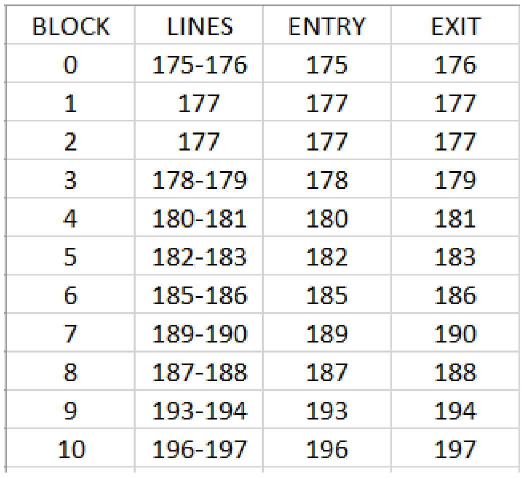
4. is_token_end(int str_com_id, int res)

Screenshot of Function:



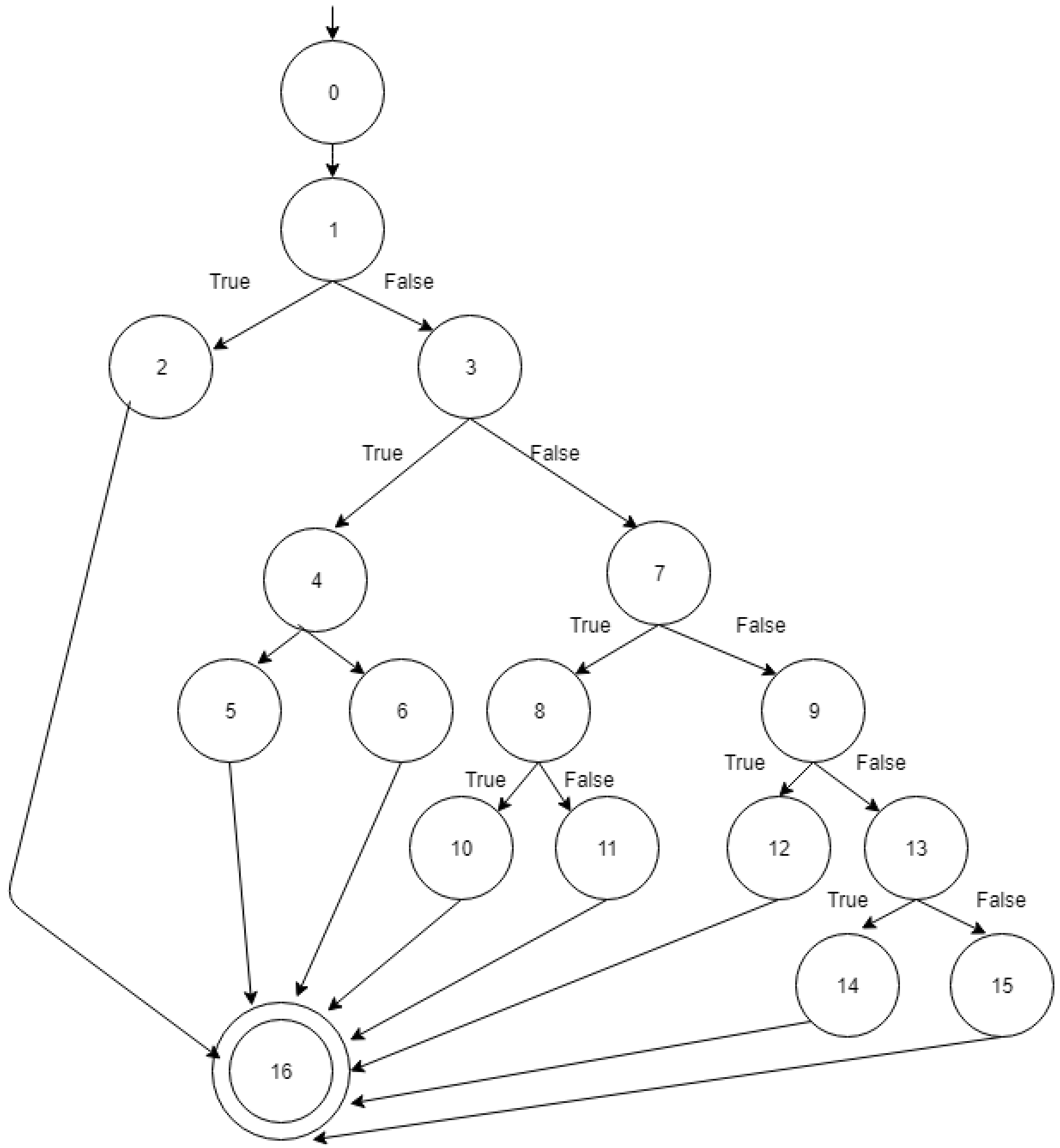
```
175       static boolean is_token_end(int str_com_id, int res)
176       {
177        if(res==1)return(true); /* is eof token? */
178        char ch = (char)res;
179        if(str_com_id==1)          /* is string token */
180           { if(ch=='"' | ch=='\n' || ch == '\r')   /* for string until meet another " */
181               return true;
182             else
183               return false;
184           }
185
186        if(str_com_id==2)     /* is comment token */
187          { if(ch=='\n' || ch == '\r' || ch==' ')      /* for comment until meet end of line */
188              return true;
189             else
190              return false;
191          }
192
193        if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
194        if(ch ==' ' || ch=='\r' || ch==59) return true;
195
196        return false;               /* other case,return FALSE */
197       }
```

Block table for the Function:



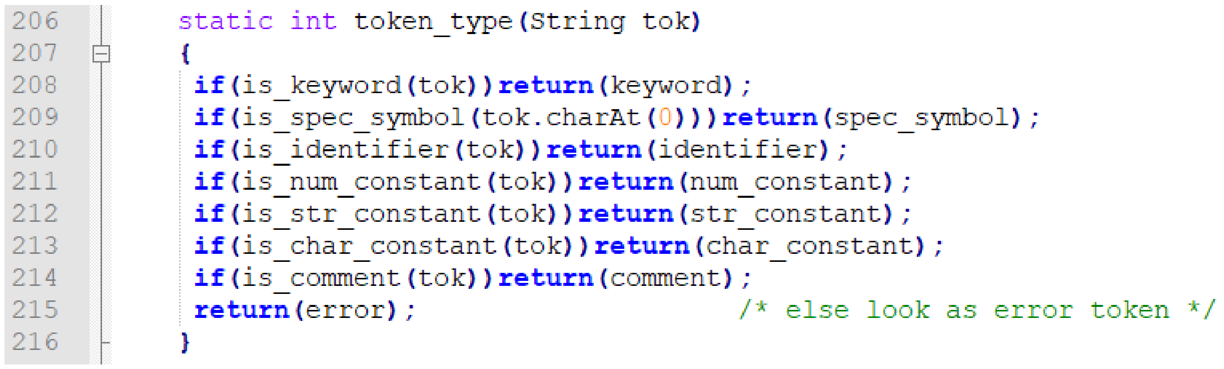
BLOCK	LINES	ENTRY	EXIT
0	175-176	175	176
1	177	177	177
2	177	177	177
3	178-179	178	179
4	180-181	180	181
5	182-183	182	183
6	185-186	185	186
7	189-190	189	190
8	187-188	187	188
9	193-194	193	194
10	196-197	196	197

Control Flow Graph for the Function:



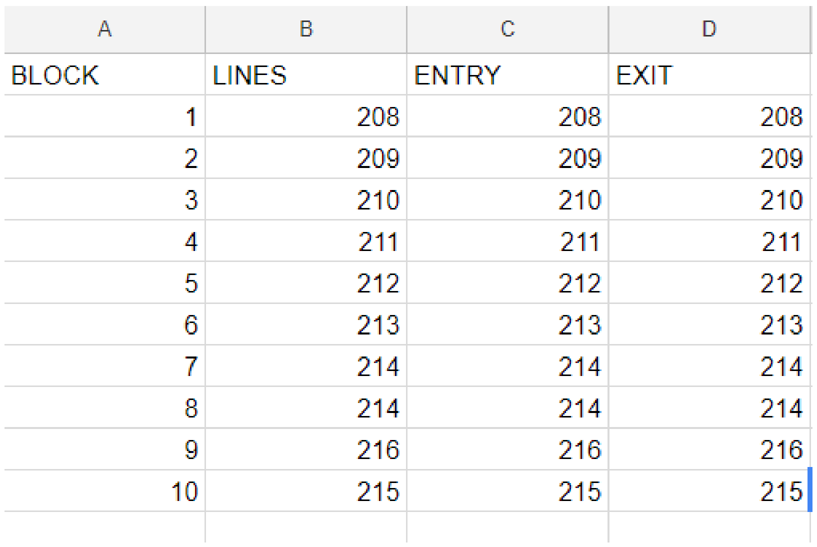
5. token_type(String tok)

Screenshot of Function:



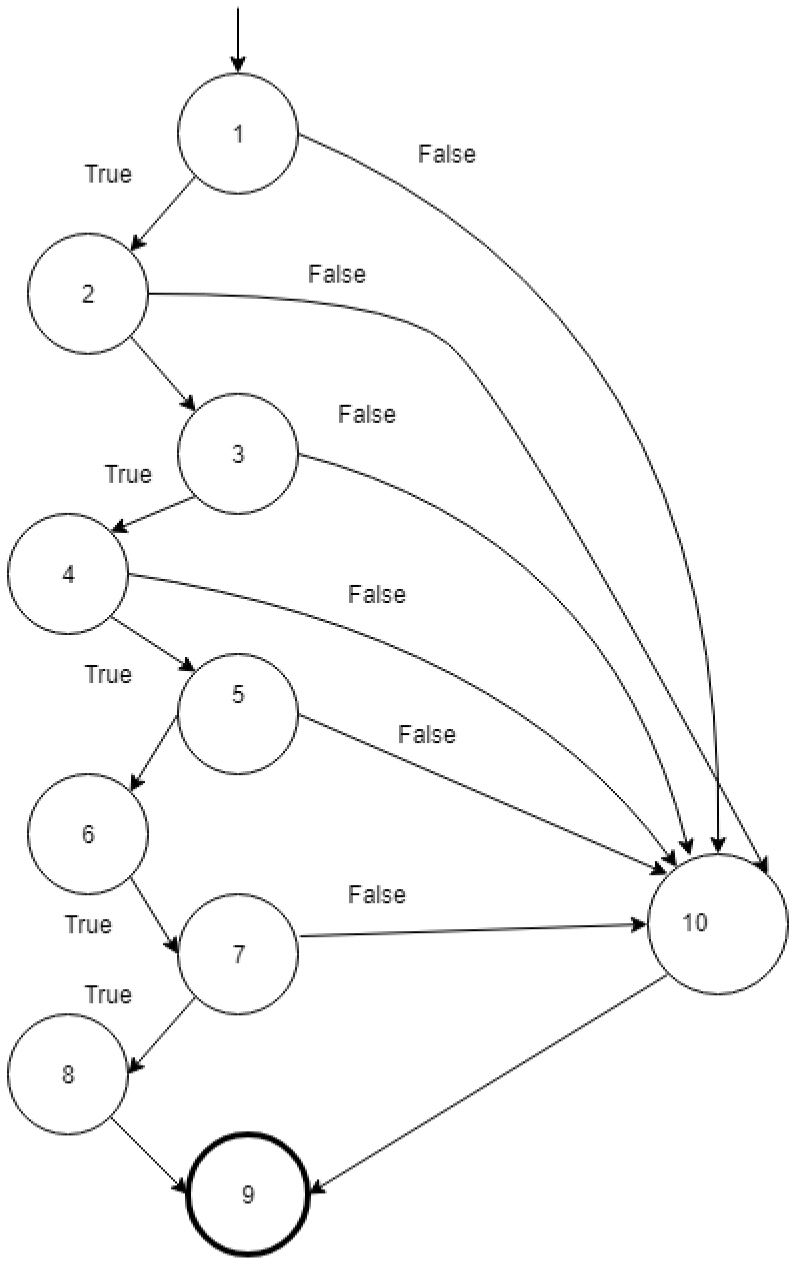
```
206     static int token_type(String tok)
207     {
208      if(is_keyword(tok))return(keyword);
209      if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
210      if(is_identifier(tok))return(identifier);
211      if(is_num_constant(tok))return(num_constant);
212      if(is_str_constant(tok))return(str_constant);
213      if(is_char_constant(tok))return(char_constant);
214      if(is_comment(tok))return(comment);
215      return(error);                          /* else look as error token */
216     }
```

Block table for the Function:



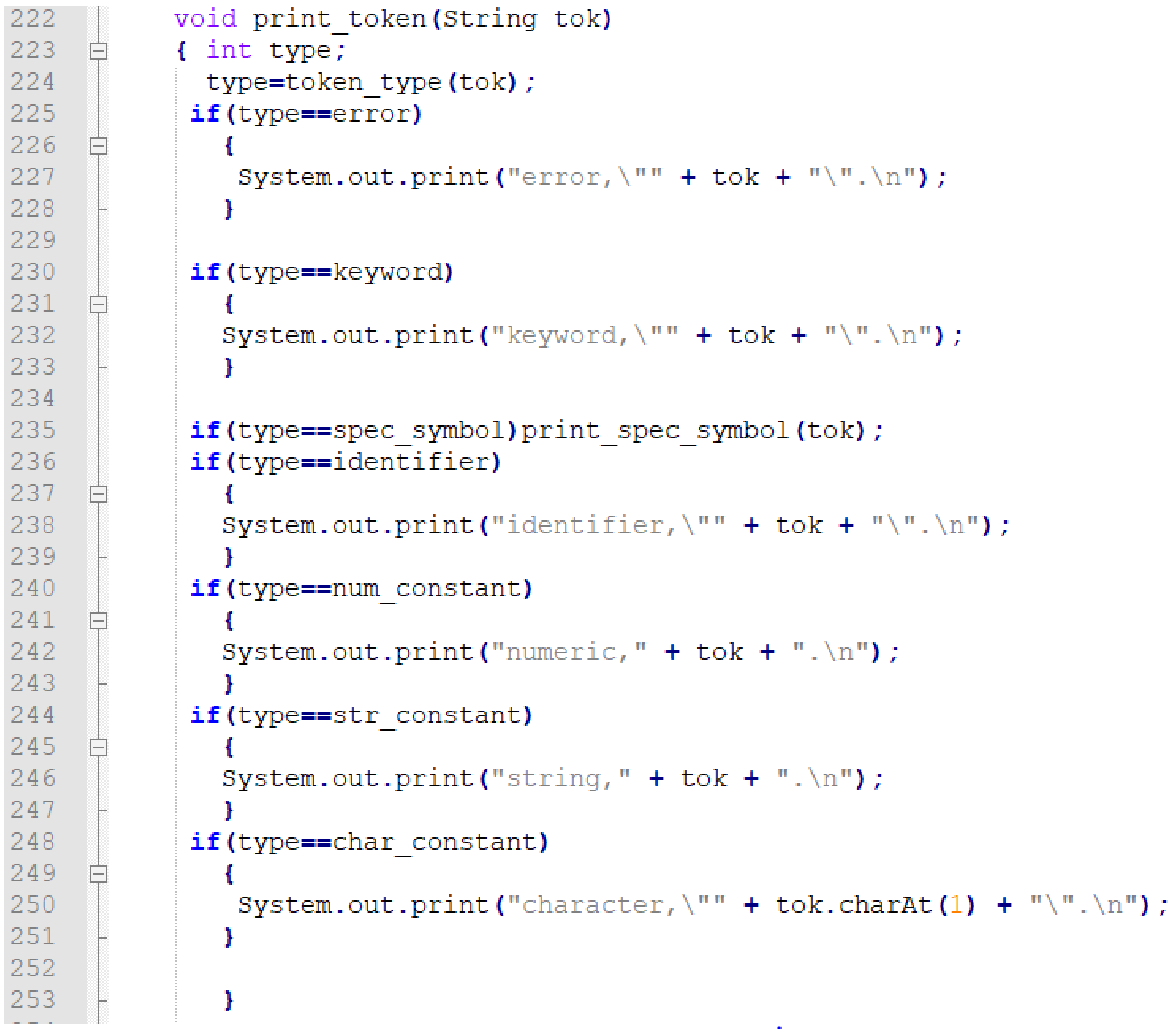
A	B	C	D
BLOCK	LINES	ENTRY	EXIT
1	208	208	208
2	209	209	209
3	210	210	210
4	211	211	211
5	212	212	212
6	213	213	213
7	214	214	214
8	214	214	214
9	216	216	216
10	215	215	215

Control Flow Graph for the Function:



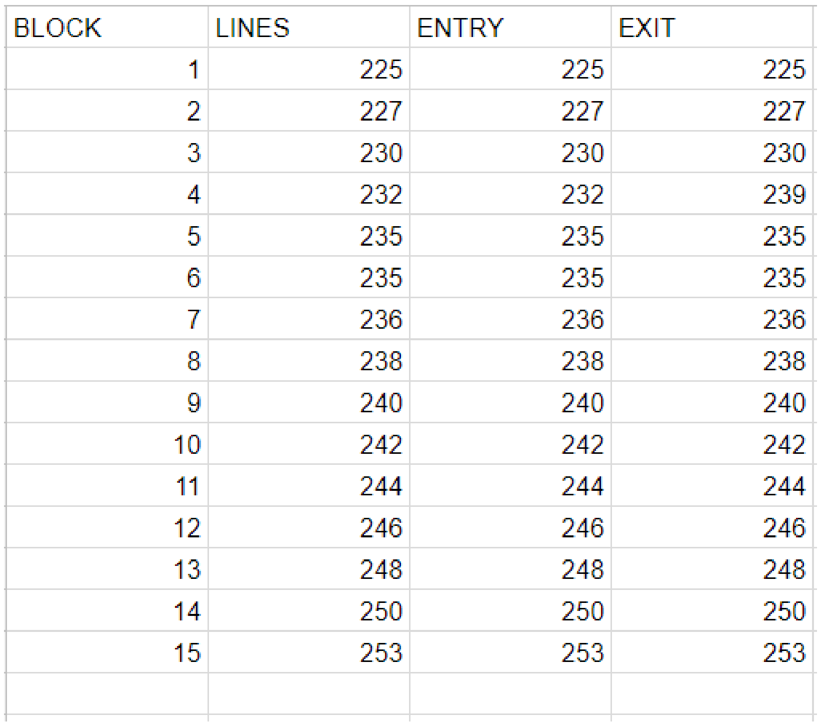
6. print_token

Screenshot of Function:



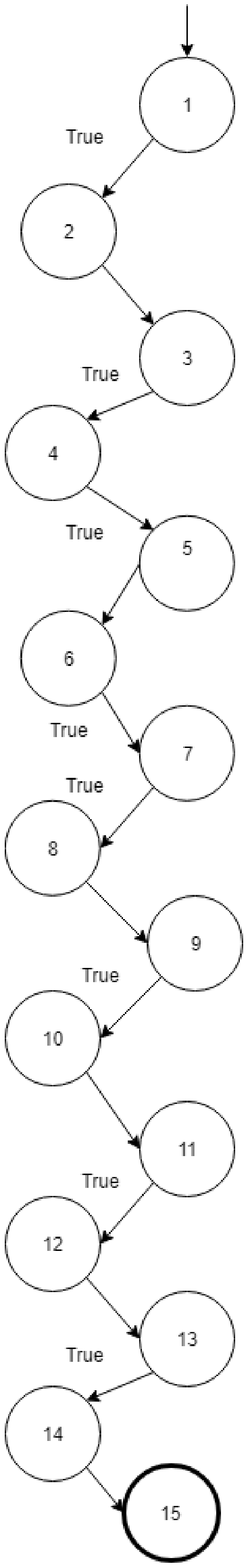
```
void print_token(String tok)
{ int type;
  type=token_type(tok);
 if(type==error)
   {
    System.out.print("error,\"" + tok + "\".\n");
   }

 if(type==keyword)
   {
   System.out.print("keyword,\"" + tok + "\".\n");
   }

 if(type==spec_symbol)print_spec_symbol(tok);
 if(type==identifier)
   {
   System.out.print("identifier,\"" + tok + "\".\n");
   }
 if(type==num_constant)
   {
   System.out.print("numeric," + tok + ".\n");
   }
 if(type==str_constant)
   {
   System.out.print("string," + tok + ".\n");
   }
 if(type==char_constant)
   {
    System.out.print("character,\"" + tok.charAt(1) + "\".\n");
   }

   }
```

Block table for the Function:



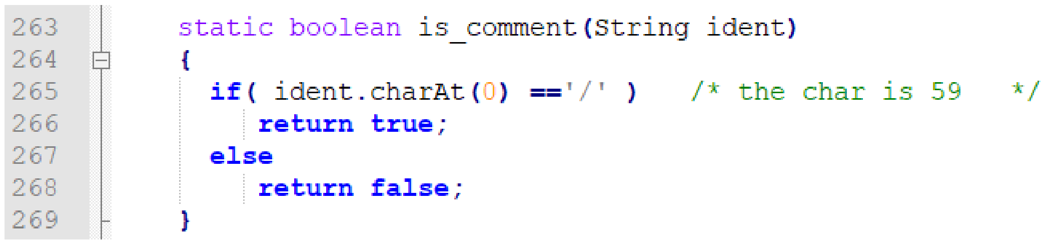
BLOCK	LINES	ENTRY	EXIT
1	225	225	225
2	227	227	227
3	230	230	230
4	232	232	239
5	235	235	235
6	235	235	235
7	236	236	236
8	238	238	238
9	240	240	240
10	242	242	242
11	244	244	244
12	246	246	246
13	248	248	248
14	250	250	250
15	253	253	253

Control Flow Graph for the Function:



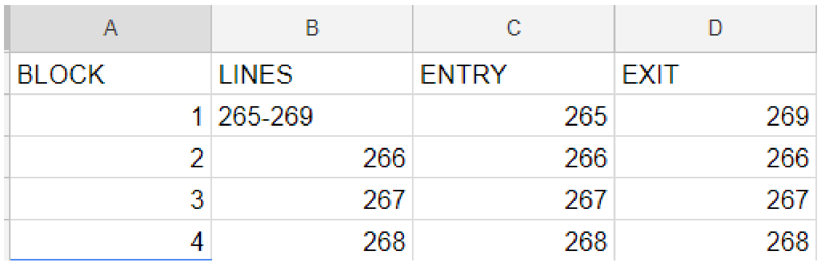
7. is_comment

Screenshot of Function:



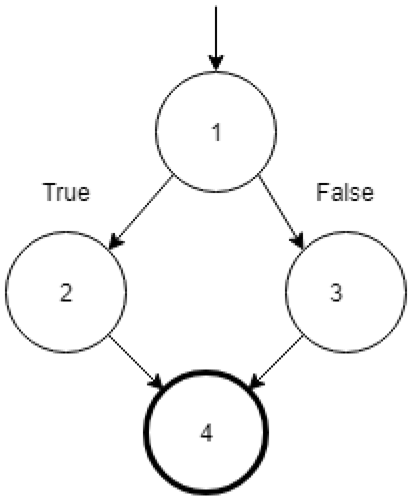
```
263        static boolean is_comment(String ident)
264        {
265          if( ident.charAt(0) =='/' )    /* the char is 59   */
266             return true;
267          else
268             return false;
269        }
```

Block table for the Function:



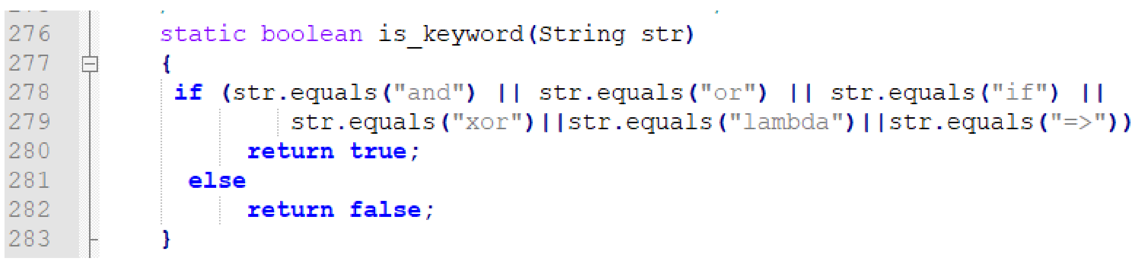
A	B	C	D
BLOCK	LINES	ENTRY	EXIT
1	265-269	265	269
2	266	266	266
3	267	267	267
4	268	268	268

Control Flow Graph for the Function:



8. is_keyword

Screenshot of Function:

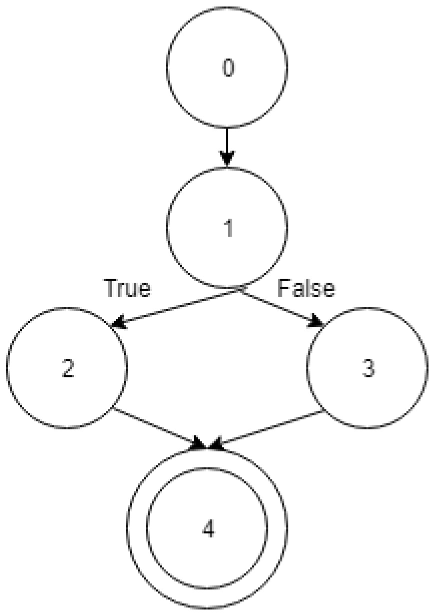


```
276        static boolean is_keyword(String str)
277        {
278         if (str.equals("and") || str.equals("or") || str.equals("if") ||
279                 str.equals("xor")||str.equals("lambda")||str.equals("=>"))
280             return true;
281          else
282             return false;
283        }
```

Block table for the Function:

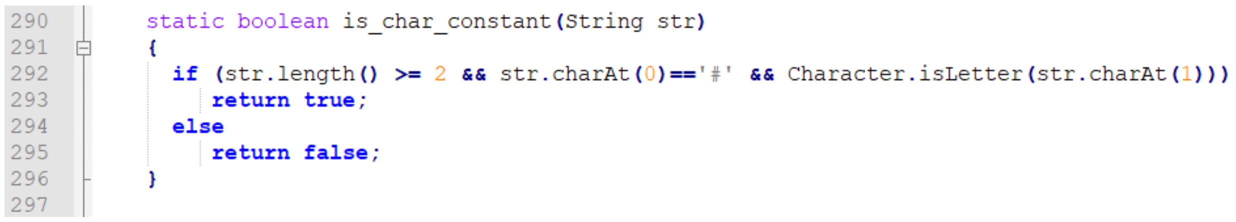
BLOCK	LINES	ENTRY	EXIT
0	276-278	276	278
1	278-279	278	279
2	280	280	280
3	281-282	281	282

Control Flow Graph for the Function:



9. is_char_constant

Screenshot of Function:

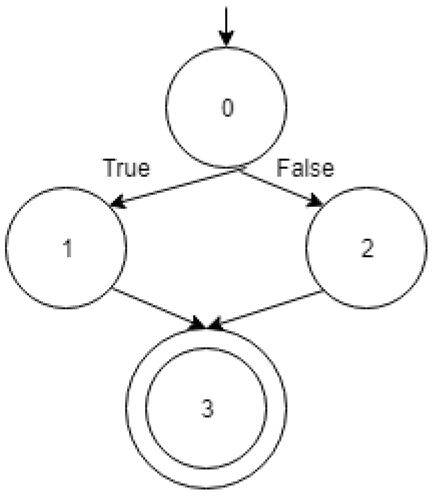


```
290     static boolean is_char_constant(String str)
291     {
292       if (str.length() >= 2 && str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
293          return true;
294       else
295          return false;
296     }
297
```

Block table for the Function:

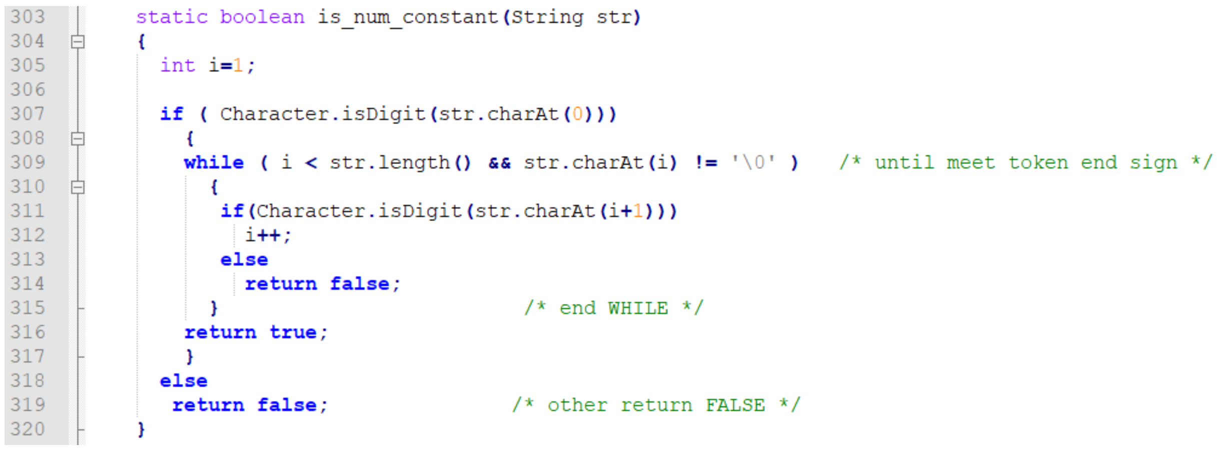
BLOCK	LINES	ENTRY	EXIT
0	290-292	290	292
1	293	293	293
2	295	295	295

Control Flow Graph for the Function:



10. is_num_constant

Screenshot of Function:

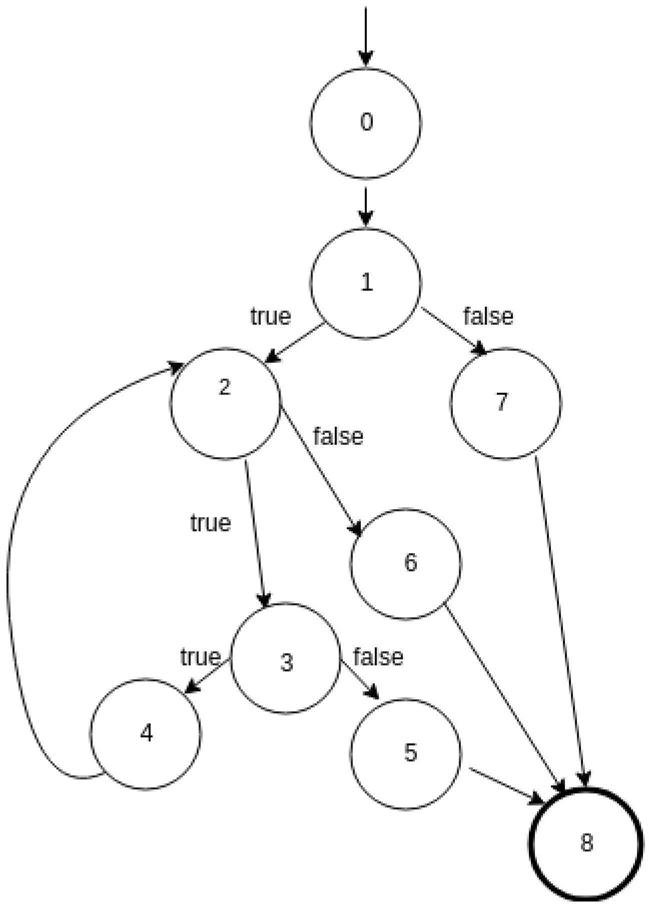


```
303  static boolean is_num_constant(String str)
304  {
305    int i=1;
306
307    if ( Character.isDigit(str.charAt(0)))
308      {
309      while ( i < str.length() && str.charAt(i) != '\0' )    /* until meet token end sign */
310        {
311         if(Character.isDigit(str.charAt(i+1)))
312           i++;
313         else
314           return false;
315        }                            /* end WHILE */
316      return true;
317      }
318    else
319     return false;                  /* other return FALSE */
320  }
```

Block table for the Function:

BLOCK	LINES	ENTRY	EXIT
0	303	303	303
1	305-307	305	307
2	309	309	309
3	311	311	311
4	312	312	312
5	314	314	314
6	316	316	316
7	319	319	319
8	320	320	320

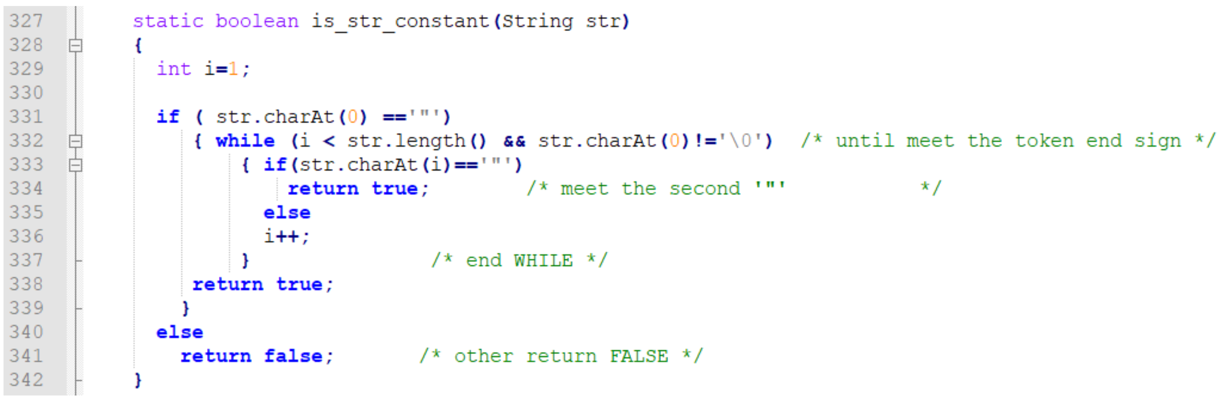
Control Flow Graph for the Function:



5

11. is_str_constant

Screenshot of Function:

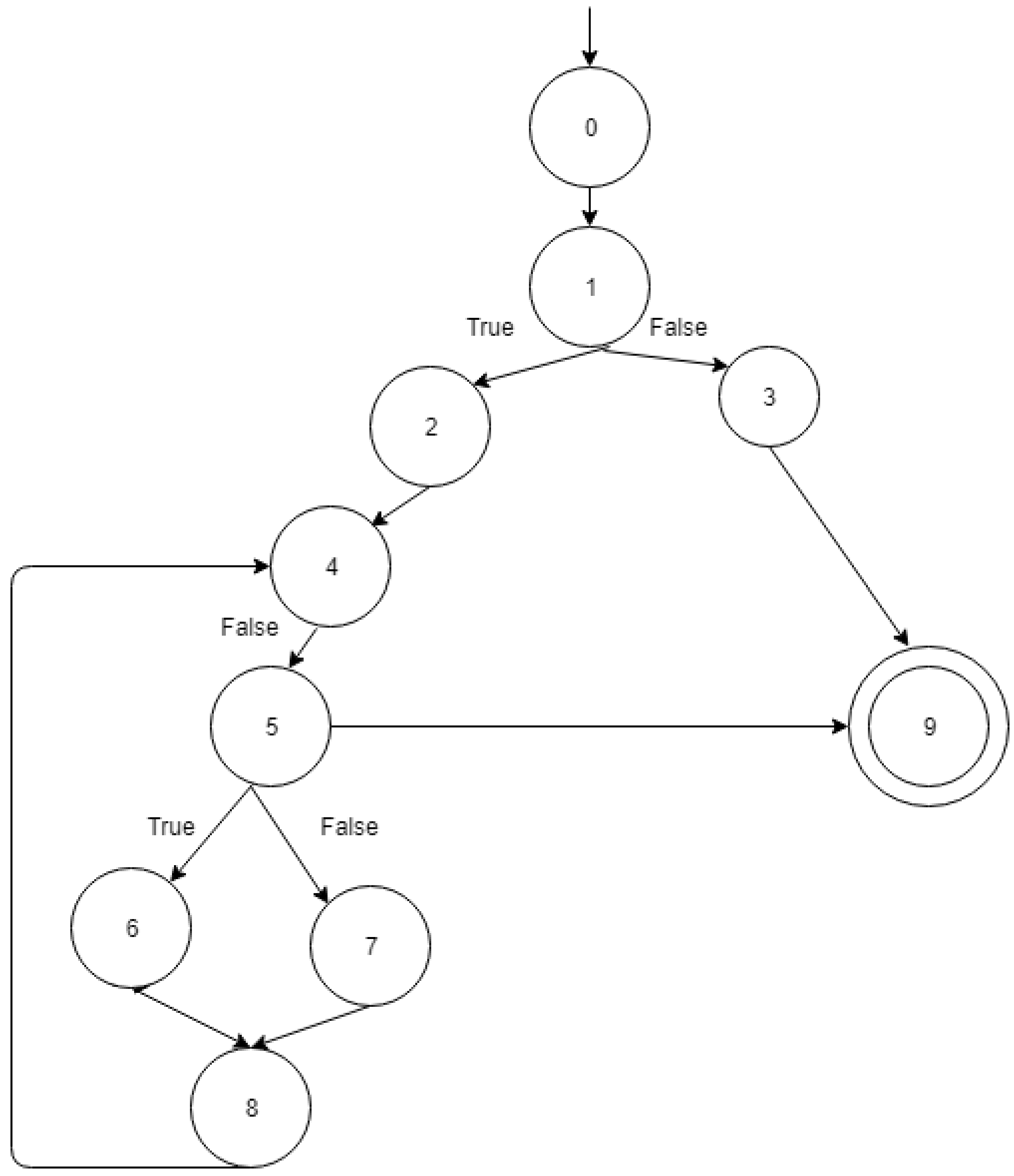


```
327      static boolean is_str_constant(String str)
328      {
329        int i=1;
330
331        if ( str.charAt(0) ==' " ')
332           { while (i < str.length() && str.charAt(0)!='\0')  /* until meet the token end sign */
333               { if(str.charAt(i)==' " ')
334                   return true;          /* meet the second '"'            */
335                 else
336                 i++;
337               }                 /* end WHILE */
338           return true;
339          }
340        else
341          return false;        /* other return FALSE */
342      }
```

Block table for the Function:

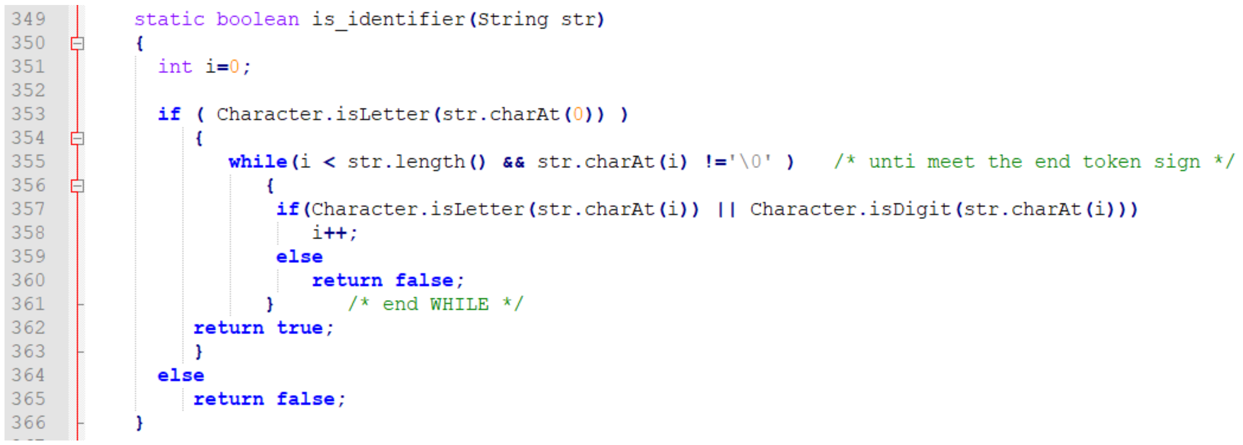
BLOCK	LINES	ENTRY	EXIT
0	327-330	327	330
1	331	331	331
2	332	332	332
3	340-341	340	341
4	332	332	332
5	333	333	333
6	334	334	334
7	336-337	336	337

Control Flow Graph for the Function:



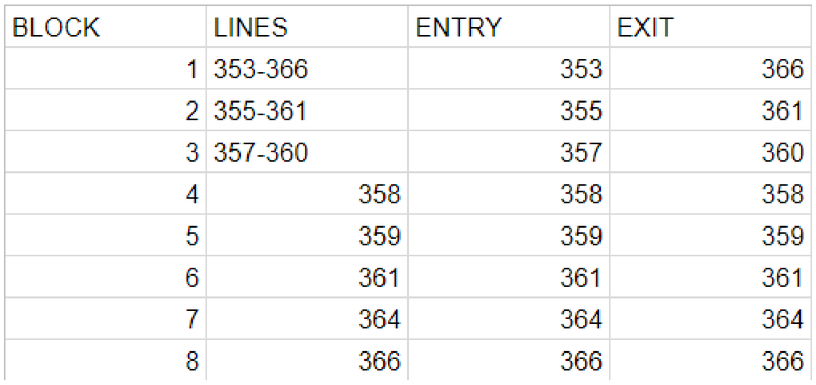
12. is_identifier

Screenshot of Function:



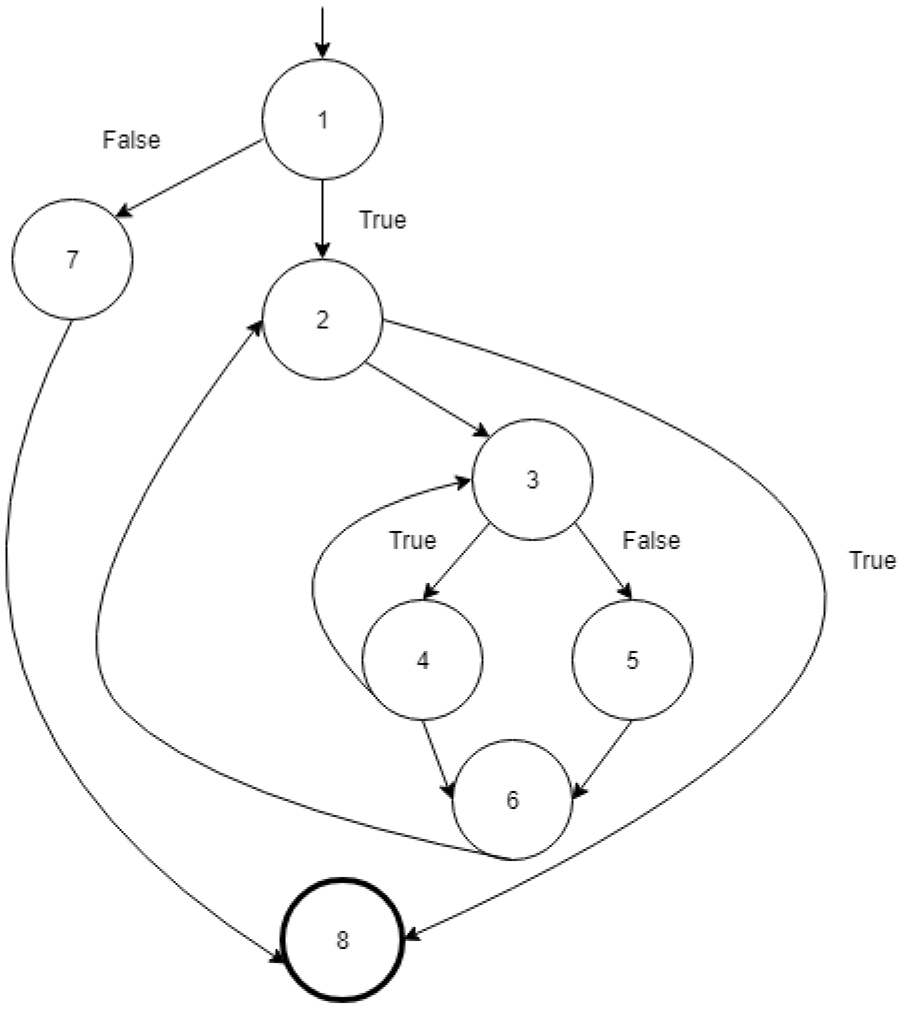
```
349     static boolean is_identifier(String str)
350     {
351       int i=0;
352
353       if ( Character.isLetter(str.charAt(0)) )
354          {
355             while(i < str.length() && str.charAt(i) !='\0' )    /* unti meet the end token sign */
356                {
357                 if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
358                    i++;
359                 else
360                    return false;
361                }      /* end WHILE */
362          return true;
363          }
364       else
365          return false;
366     }
```

Block table for the Function:



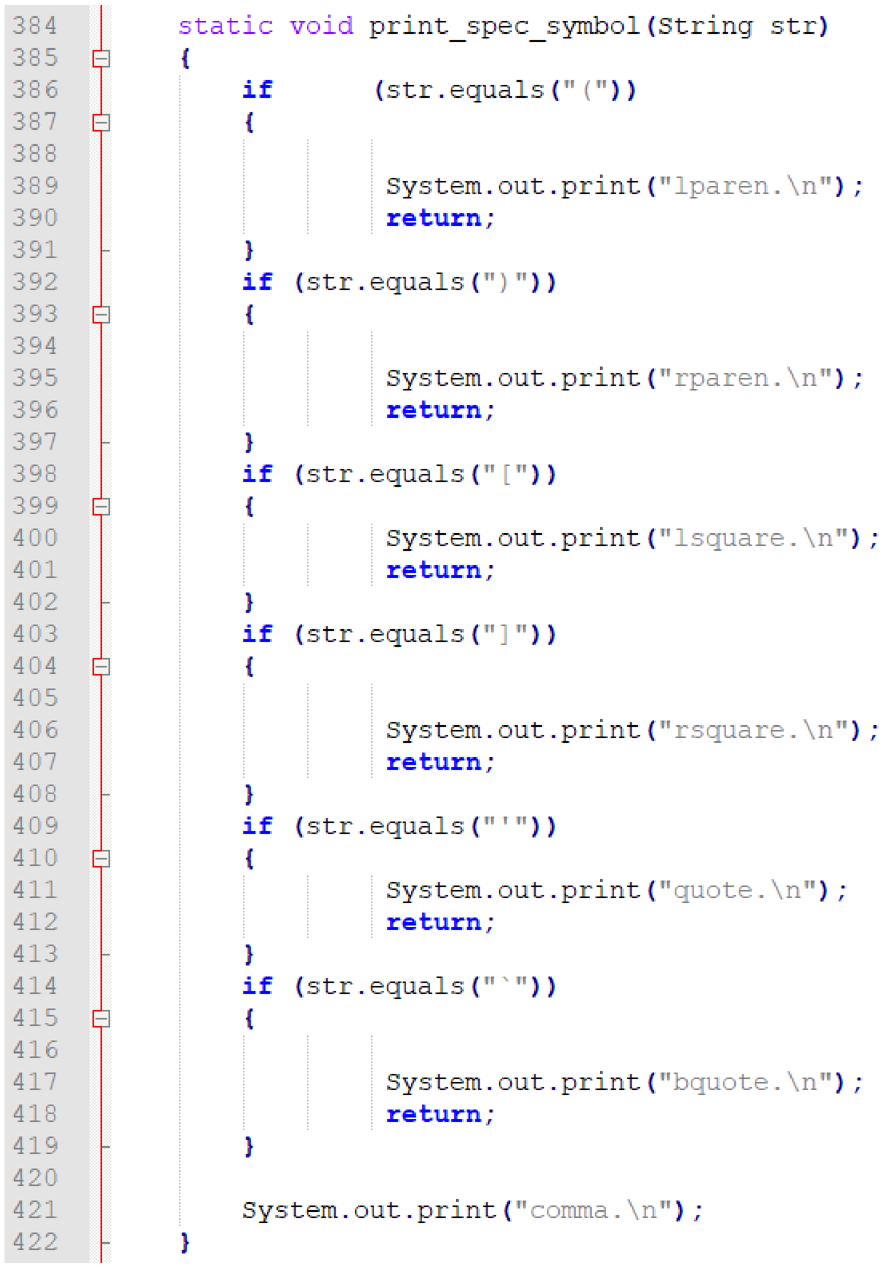
BLOCK	LINES	ENTRY	EXIT
1	353-366	353	366
2	355-361	355	361
3	357-360	357	360
4	358	358	358
5	359	359	359
6	361	361	361
7	364	364	364
8	366	366	366

Control Flow Graph for the Function:



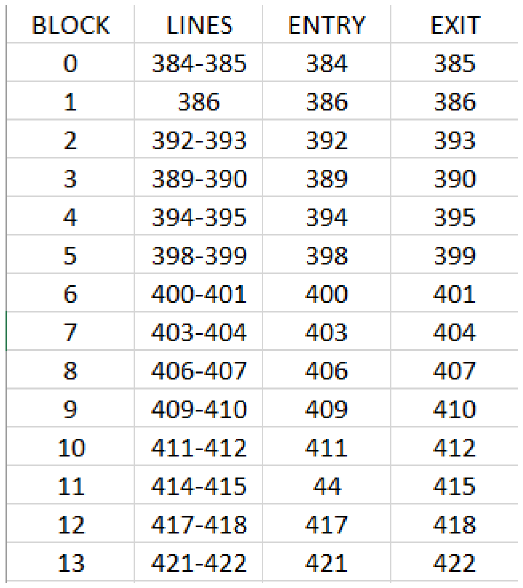
13. print_spec_symbol

Screenshot of Function:



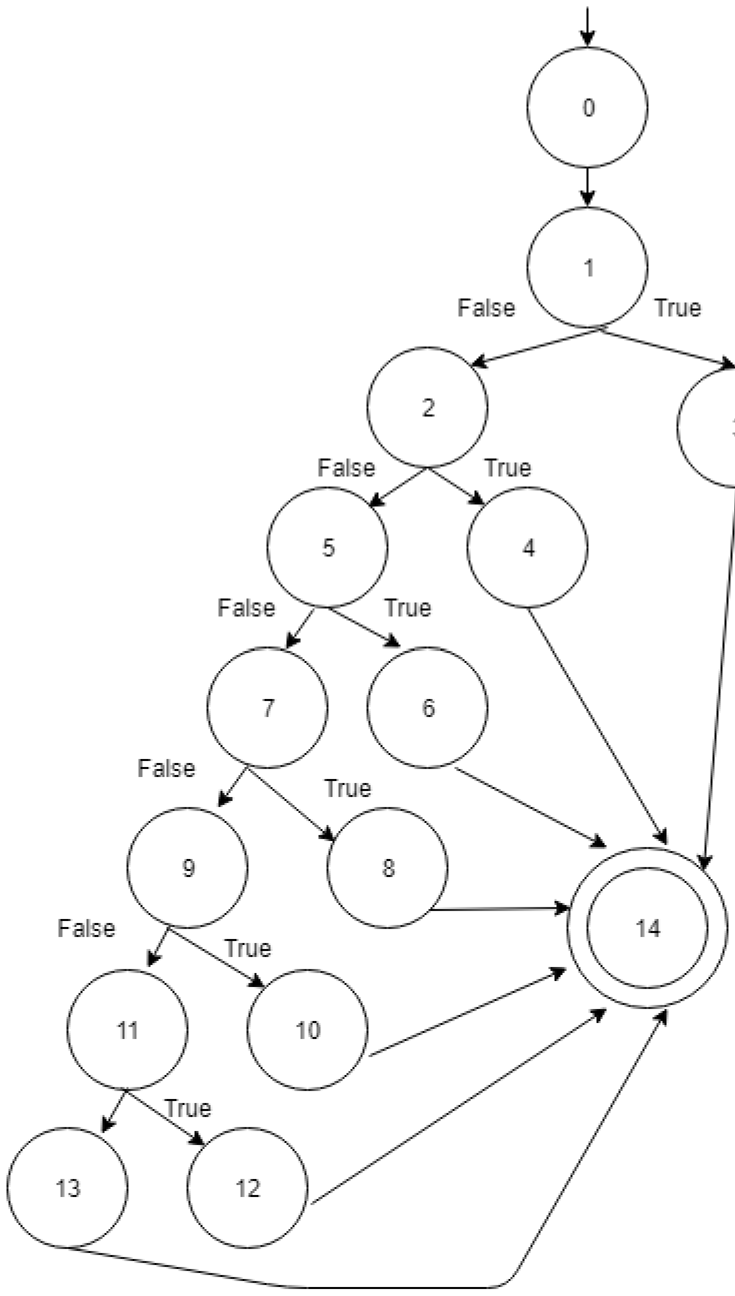
```
static void print_spec_symbol(String str)
{
    if      (str.equals("("))
    {

            System.out.print("lparen.\n");
            return;
    }
    if (str.equals(")"))
    {

            System.out.print("rparen.\n");
            return;
    }
    if (str.equals("["))
    {
            System.out.print("lsquare.\n");
            return;
    }
    if (str.equals("]"))
    {

            System.out.print("rsquare.\n");
            return;
    }
    if (str.equals("'"))
    {
            System.out.print("quote.\n");
            return;
    }
    if (str.equals("`"))
    {

            System.out.print("bquote.\n");
            return;
    }

    System.out.print("comma.\n");
}
```

Block table for the Function:



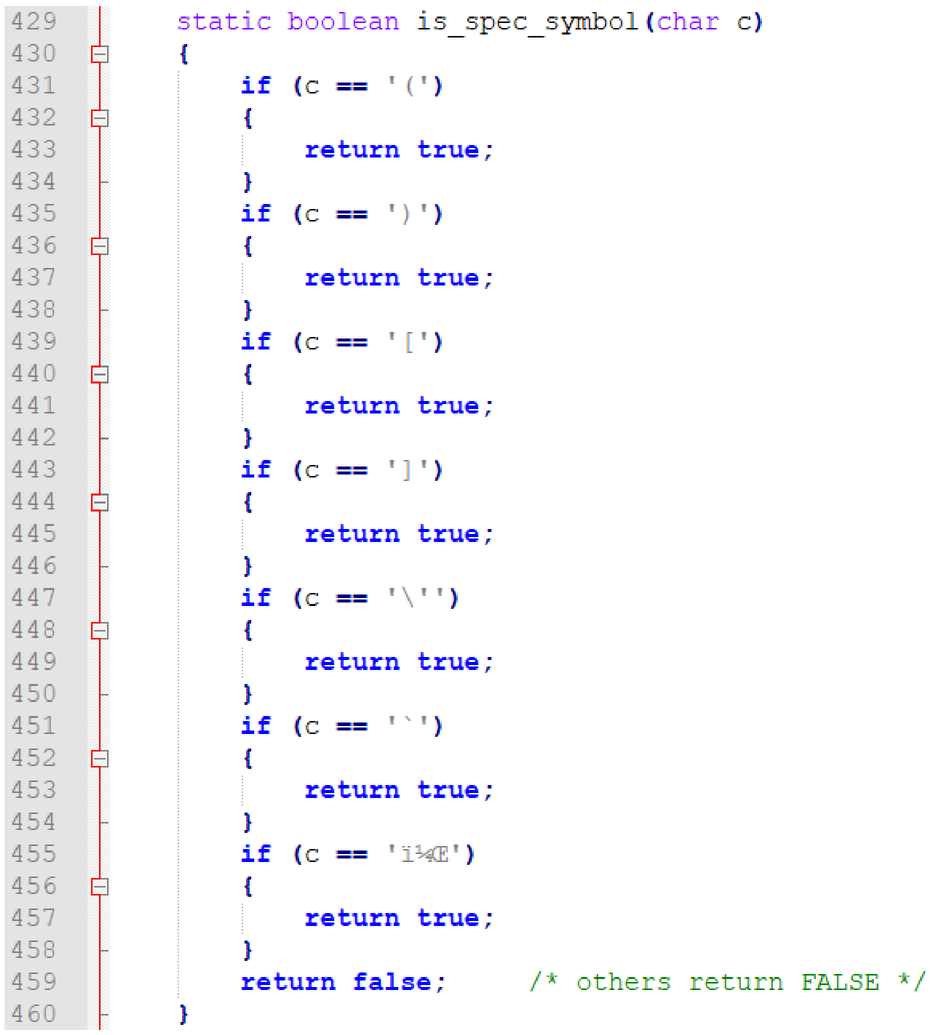
BLOCK	LINES	ENTRY	EXIT
0	384-385	384	385
1	386	386	386
2	392-393	392	393
3	389-390	389	390
4	394-395	394	395
5	398-399	398	399
6	400-401	400	401
7	403-404	403	404
8	406-407	406	407
9	409-410	409	410
10	411-412	411	412
11	414-415	44	415
12	417-418	417	418
13	421-422	421	422

Control Flow Graph for the Function:



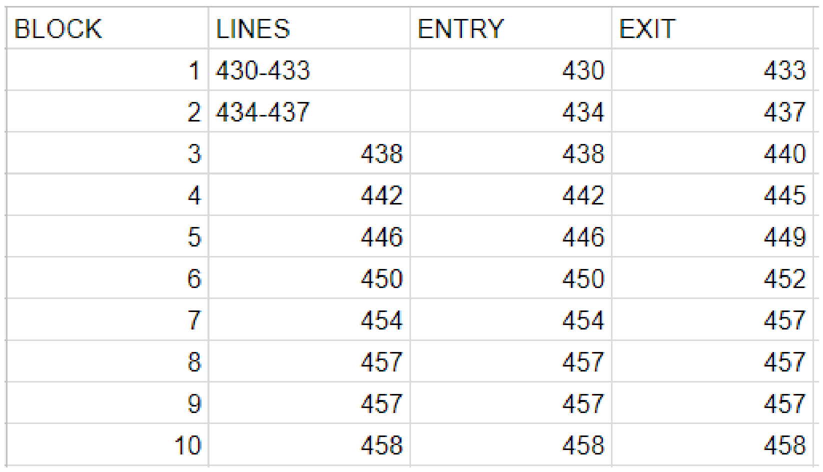
14. is_spec_symbol

Screenshot of Function:



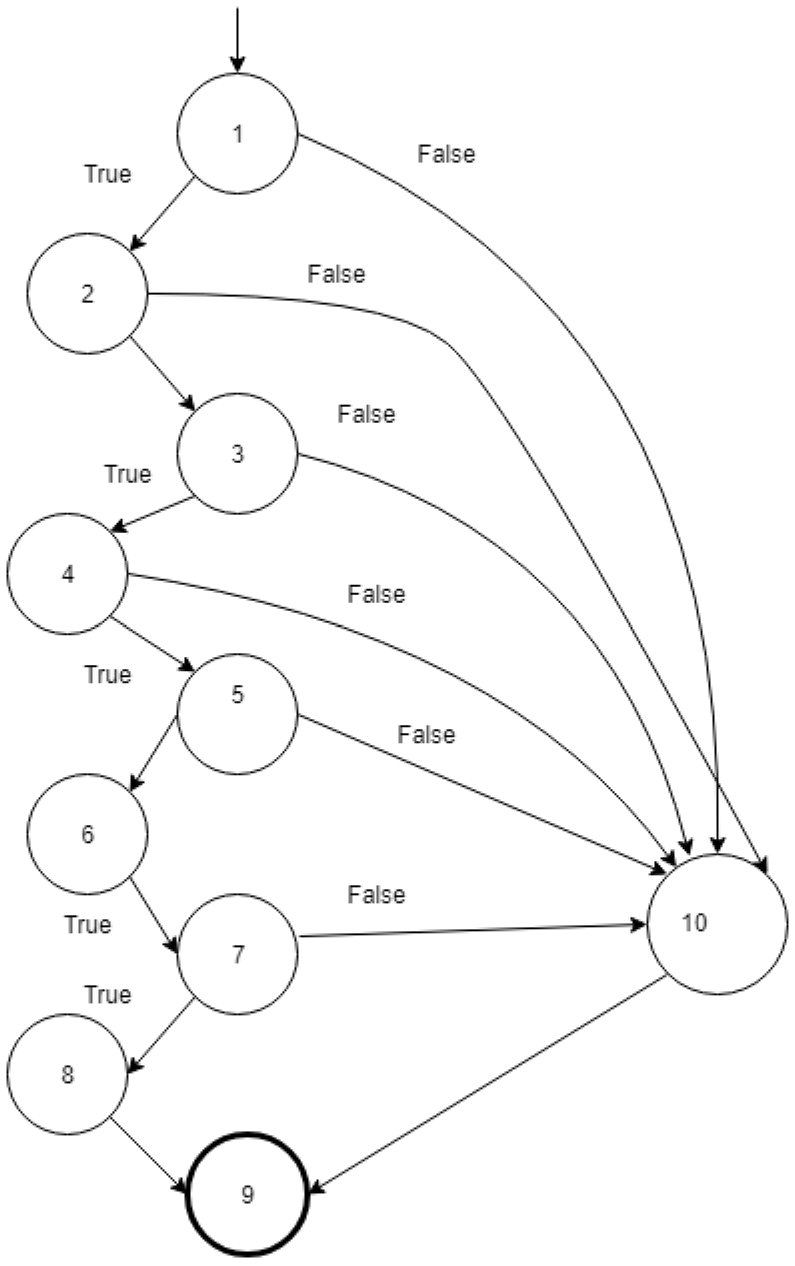
```c
static boolean is_spec_symbol(char c)
{
    if (c == '(')
    {
        return true;
    }
    if (c == ')')
    {
        return true;
    }
    if (c == '[')
    {
        return true;
    }
    if (c == ']')
    {
        return true;
    }
    if (c == '\'')
    {
        return true;
    }
    if (c == '`')
    {
        return true;
    }
    if (c == 'ï¼Œ')
    {
        return true;
    }
    return false;      /* others return FALSE */
}
```

Block table for the Function:



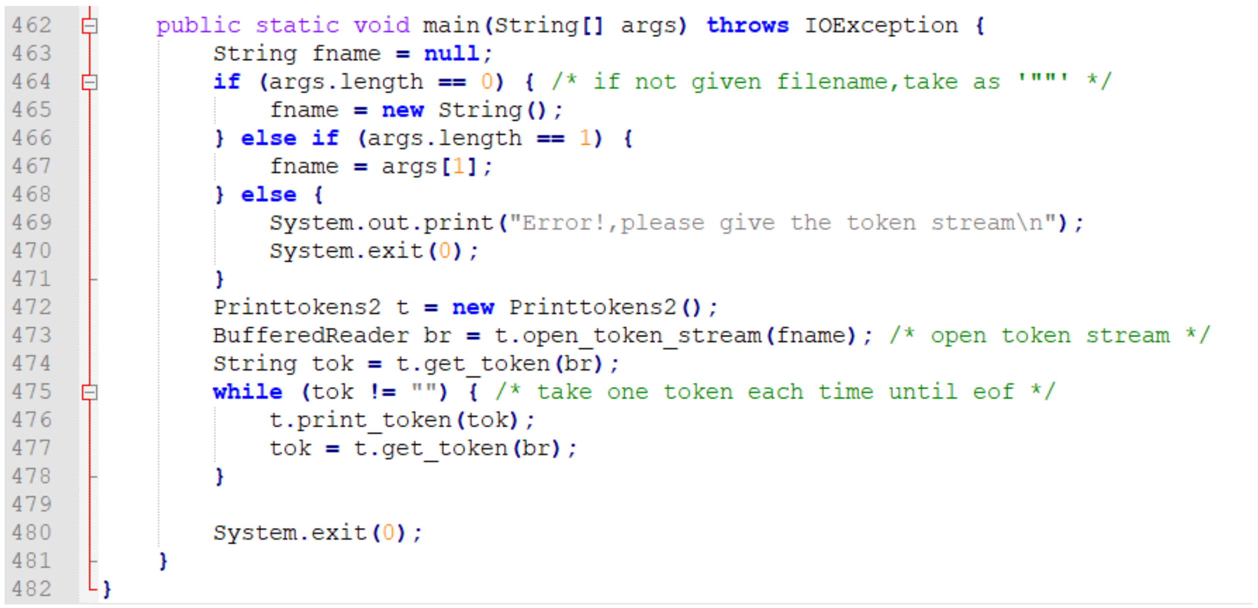
BLOCK	LINES	ENTRY	EXIT
1	430-433	430	433
2	434-437	434	437
3	438	438	440
4	442	442	445
5	446	446	449
6	450	450	452
7	454	454	457
8	457	457	457
9	457	457	457
10	458	458	458

Control Flow Graph for the Function:



15. Main

Screenshot of Function:

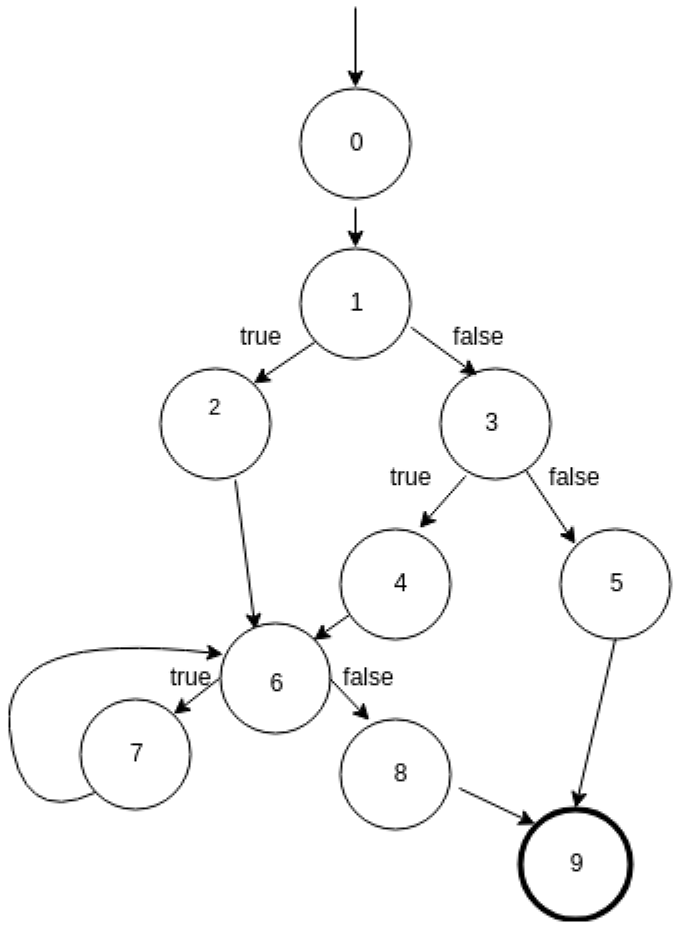


```
462 	public static void main(String[] args) throws IOException {
463 		String fname = null;
464 		if (args.length == 0) { /* if not given filename,take as '""' */
465 			fname = new String();
466 		} else if (args.length == 1) {
467 			fname = args[1];
468 		} else {
469 			System.out.print("Error!,please give the token stream\n");
470 			System.exit(0);
471 		}
472 		Printtokens2 t = new Printtokens2();
473 		BufferedReader br = t.open_token_stream(fname); /* open token stream */
474 		String tok = t.get_token(br);
475 		while (tok != "") { /* take one token each time until eof */
476 			t.print_token(tok);
477 			tok = t.get_token(br);
478 		}
479 
480 		System.exit(0);
481 	}
482 }
```

Block table for the Function:

BLOCK	LINES	ENTRY	EXIT
0	462	462	462
1	463-464	463	464
2	465	465	465
3	466	466	466
4	467	467	467
5	468-470	468	470
6	472-475	472	475
7	476-477	476	477
8	480	480	480
9	481	481	481

Control Flow Graph for the Function:



By
Rithin Surya Sai Nadh Gadapa(1001565680)
Haritha Soundararaj(1001624272)
Nandan Pandya(1001626050)