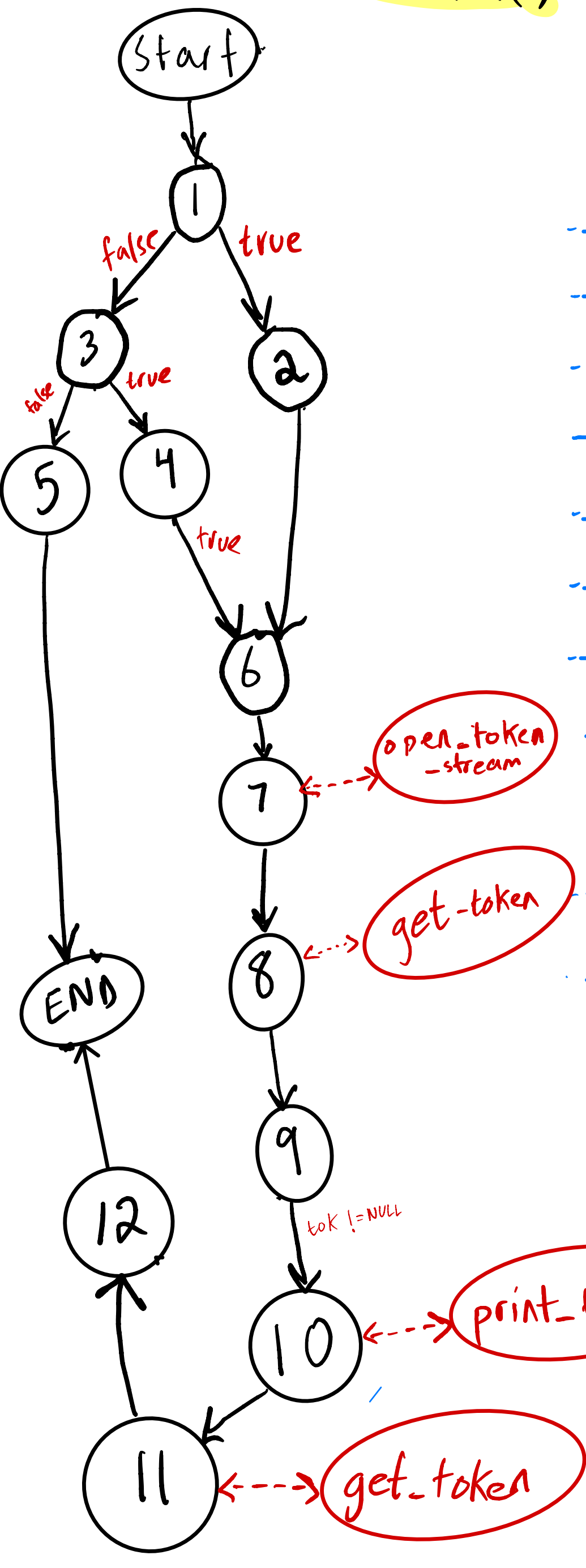
```
457
458 ▶ @     public static void main(String[] args)  {
459             String fname = null;
460             if (args.length == 0) { /* if not given filename,take as '"' */
461                 fname = new String();
462             } else if (args.length == 1) {
463                 fname = args[0];
464             } else {
465                 System.out.print("Error! Please give the token stream\n");
466                 System.exit( status: 0);
467             }
468             Printtokens t = new Printtokens();
469             BufferedReader br = t.open_token_stream(fname); /* open token stream */
470             String tok = t.get_token(br);
471             while (tok != null) {   /* take one token each time until eof */
472                 t.print_token(tok);
473                 tok = t.get_token(br);
474
475             }
476
477             System.exit( status: 0);
478         }
479     }
```

main()

CFG's



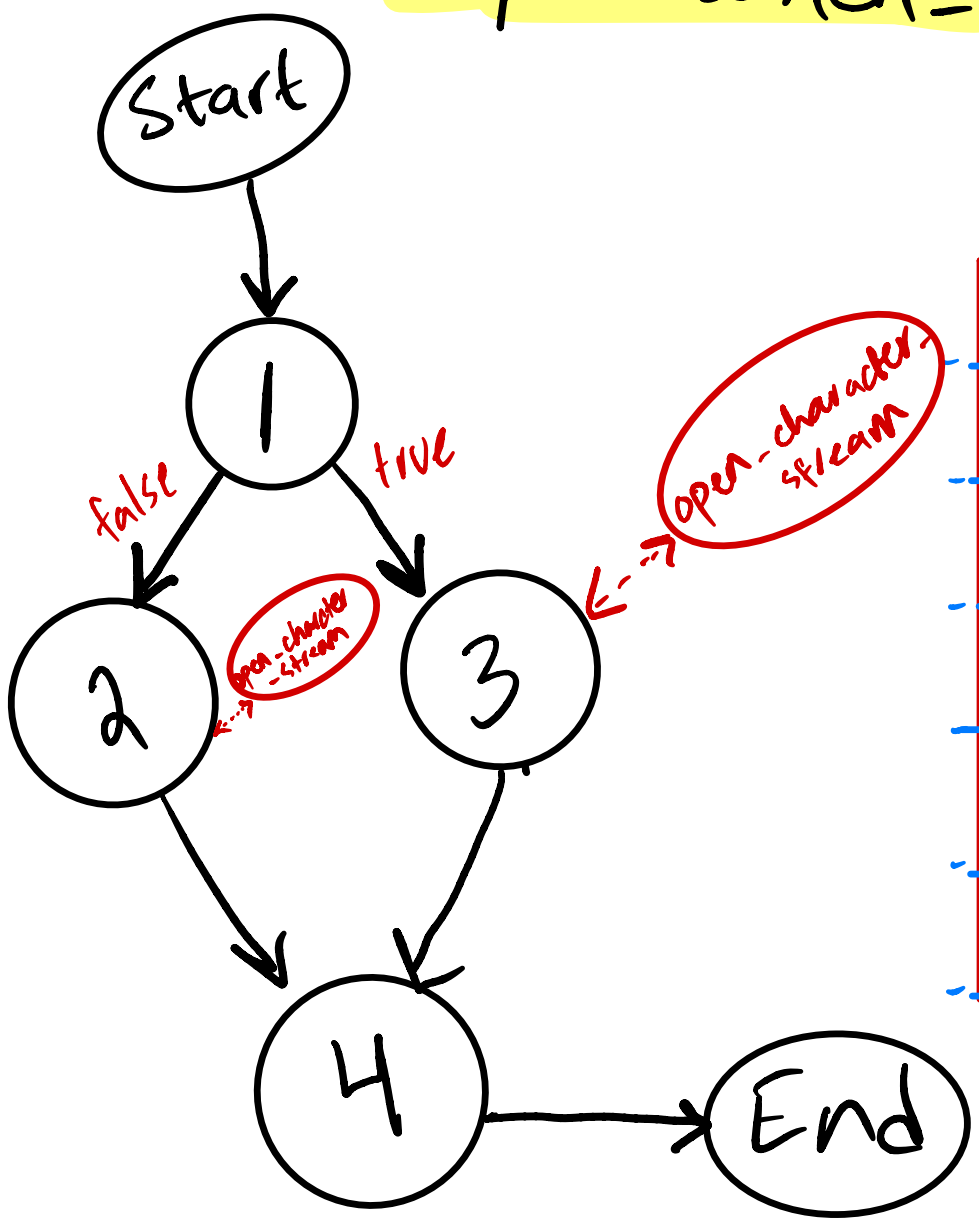
Block Number	Lines	Entry	Exit	Function Call
1	459, 460	459	460	
2	461	461	461	
3	462	462	462	
4	463	463	463	
5	464	464	464	
6	468	468	468	
7	469	469	469	open_token_stream
8	470	470	470	get_token
9	471	471	471	
10	472	472	472	print_token
11	473	473	473	get_token
12	477	477	477	

```
/*******************************************************/
1 usage
BufferedReader open_token_stream(String fname)
{
    BufferedReader br;
if(fname==null || fname.equals(""))
    br=open_character_stream( fname: null);
 else
    br=open_character_stream(fname);
 return br;
}

/*******************************************************/
/* NAME :   get_token                                  */
/* INPUT:   a BufferedReader          */
/* OUTPUT:      a token string                                 */
/* DESCRIPTION: according the syntax of tokens,dealing  */
/*              with different case  and get one token  */
/*******************************************************/
2 usages
```

```
2 usages
BufferedReader open_character_stream(String fname) {
    BufferedReader br = null;
    if (fname == null) {
        br = new BufferedReader(new InputStreamReader(System.in));
    } else {
        try {
            FileReader fr = new FileReader(fname);
            br = new BufferedReader(fr);
        } catch (FileNotFoundException e) {
            System.out.print("The file " + fname +" doesn't exists\n");
            e.printStackTrace();
        }
    }

    return br;
}
```

open_token_stream

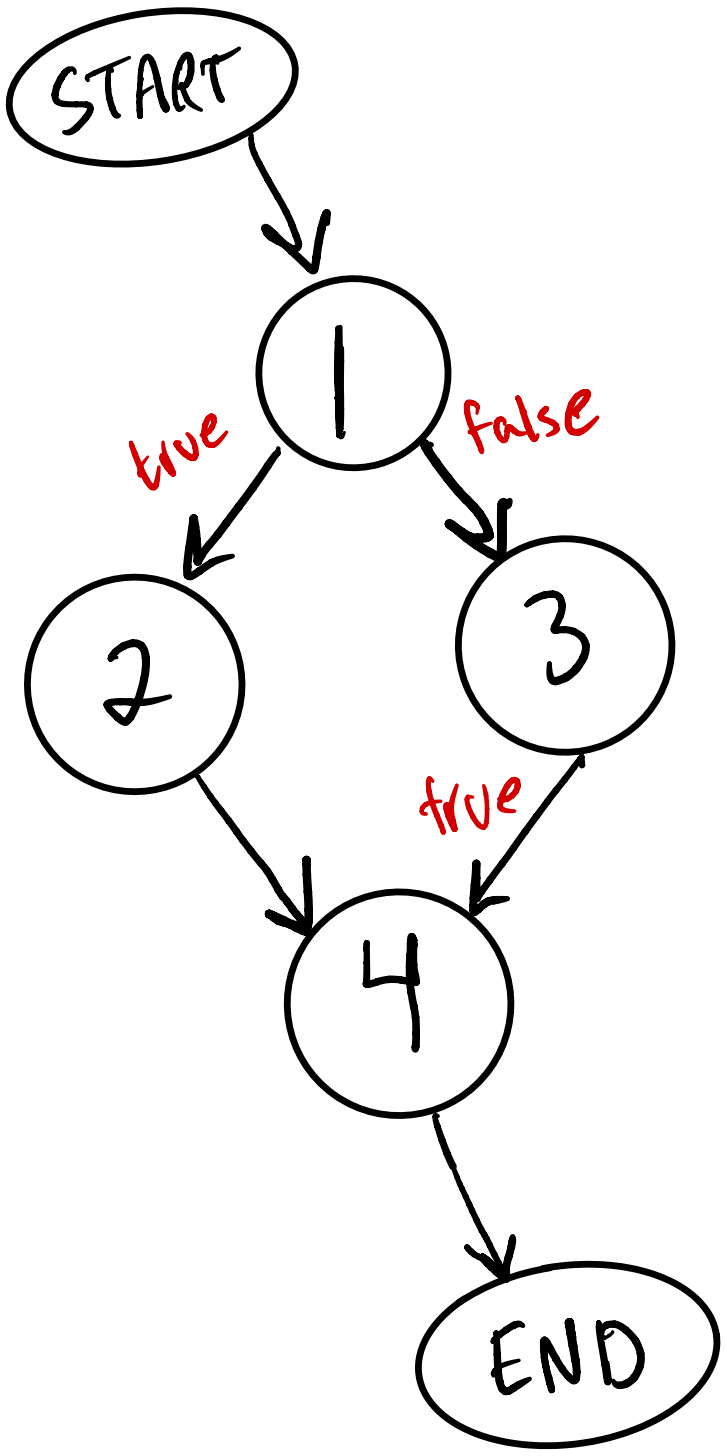
☆ main() function



Block Number	Lines	Entry	Exit	Function Call
1	79,80	79	80	
2	81	81	81	open_character_stream
3	83	83	83	open_character_stream
4	84	84	84	
5				

---

open_character_stream



Block Number	Lines	Entry	Exit	Function Call
1	23,24	23	24	
2	25	25	25	
3	28,29	28	29	
4	36	36	36	
5				
6				

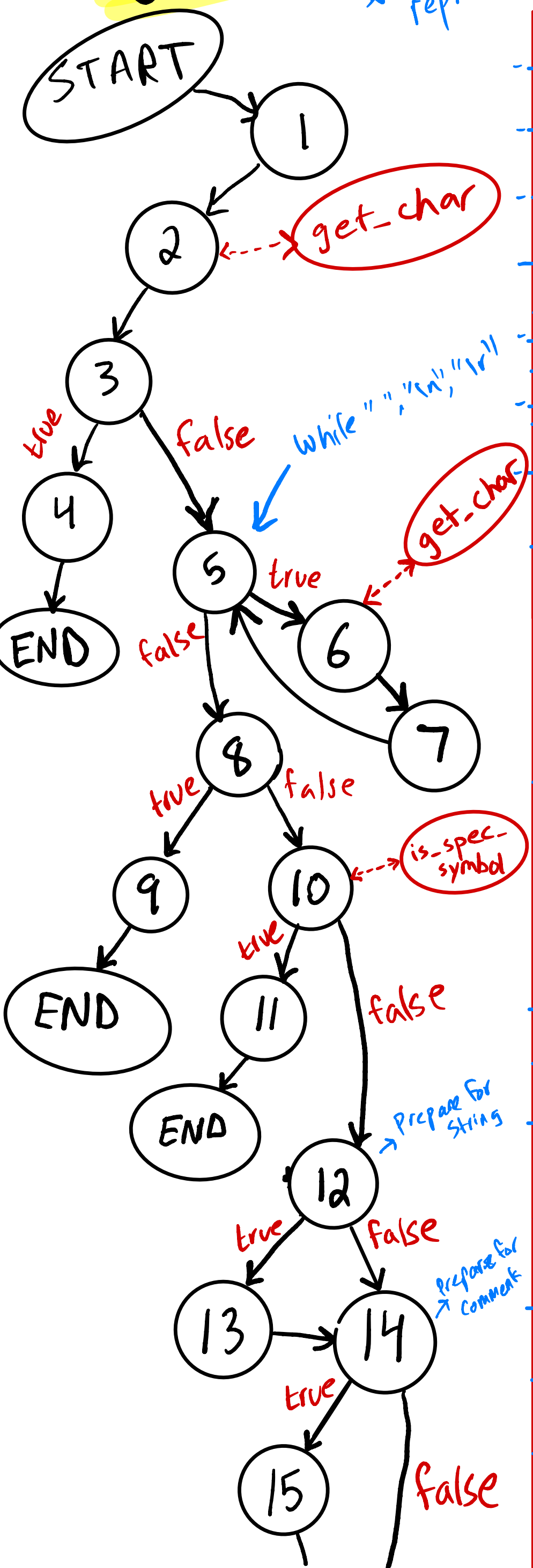
```
2 usages
String get_token(BufferedReader br)
{
  int i=0,j;
  int id=0;
  int res = 0;
  char ch = '\0';

  StringBuilder sb = new StringBuilder();

  try {
      res = get_char(br);
      if (res == -1) {
          return null;
      }
      ch = (char)res;
     while(ch==' '||ch=='\n' || ch == '\r')
     {
       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=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;
```

```
127
128         while (is_token_end(id,res) == false)/* until meet the end character */
129         {
130             sb.append(ch);
131             br.mark( readAheadLimit: 4);
132             res = get_char(br);
133             if (res == -1) {
134                 break;
135             }
136             ch = (char)res;
137         }
138
139         if(res == -1)       /* if end character is eof token    */
140            { unget_char(ch,br);        /* then put back eof on token_stream */
141              return sb.toString();
142            }
143
144         if(is_spec_symbol(ch)==true)     /* if end character is special_symbol */
145            { unget_char(ch,br);        /* then put back this character       */
146              return sb.toString();
147            }
148         if(id==1)                  /* if end character is " and is string */
149           {
150             if (ch == '"') {
151                 sb.append(ch);
152             }
153             return sb.toString();
154           }
155         if(id==0 && ch==59)
156                                       /* when not in string or comment,meet ";" */
157           { unget_char(ch,br);       /* then put back this character         */
158             return sb.toString();
159           }
160     } catch (IOException e) {
161         e.printStackTrace();
162     }
163
164         return sb.toString();                   /* return nomal case token            */
165     }
```

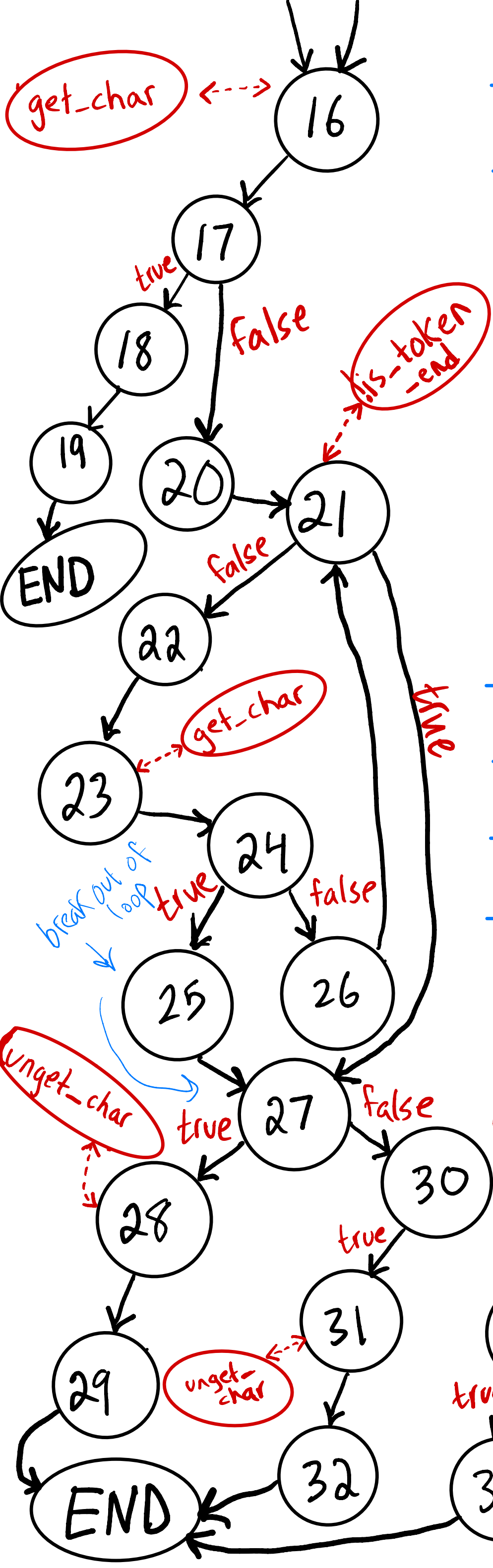
get_token

★multiple End Nodes represent the same End Node

★main() function



Block Number	Lines	Entry	Exit	Function Call
1	96,97,98,99, 101	96	101	
2	104	104	104	get_char
3	105	105	105	
4	106	106	106	
5	108,109	108	109	
6	111	111	111	get_char
7	112	112	112	
8	115a	115a	115a	
9	115b	115b		
10	116,117a	116	117a	is_spec_symbol -true
11	117b	117b	117b	
12	118a	118a	118a	
13	118b	118b	118b	
14	119a	119a	119a	
15	119b	119b	119b	
16	121	121	121	get_char
17	122	122	122	
18	123	123	123	unget_char
19	124	124	124	
20	126	126	126	
21	128	128	128	is_token_end
22	130,131	130	131	
23	132	132	132	get_char
24	133	133	133	
25	134	134	134	
26	136,128	136	128	
27	139	139	139	

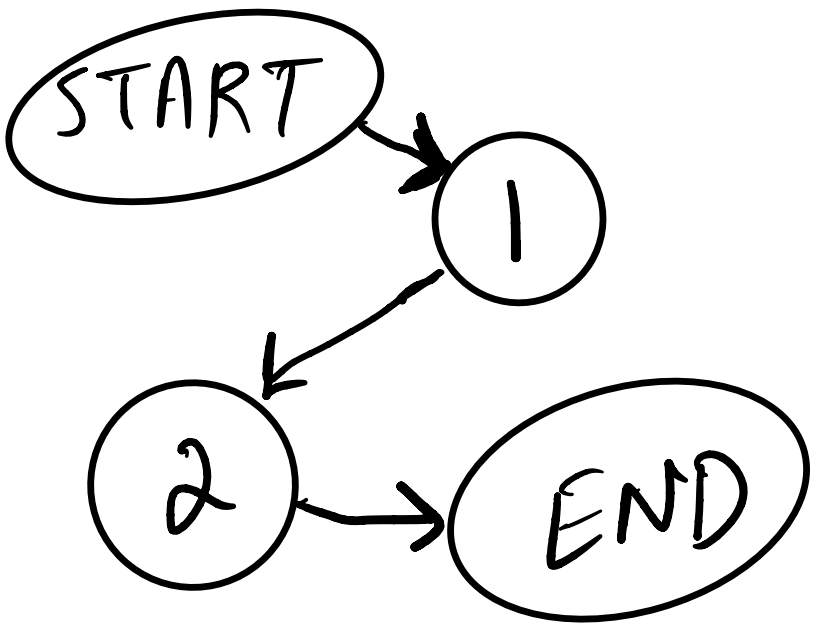


28	140	140	140	unget_char
29	141	141	141	
30	144	144	144	is_spec_symbol
31	145	145	145	unget_char
32	146	146	146	
33	148	148	148	
34	150	150	150	
35	151,153	151	153	
36	155	155	155	
37	157	157	157	unget_char
38	158	158	158	
39	164	164	164	

```
4 usages
int get_char(BufferedReader br){
    int ch = 0;
    try {
        br.mark( readAheadLimit: 4);
        ch= br.read();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return ch;
}
```

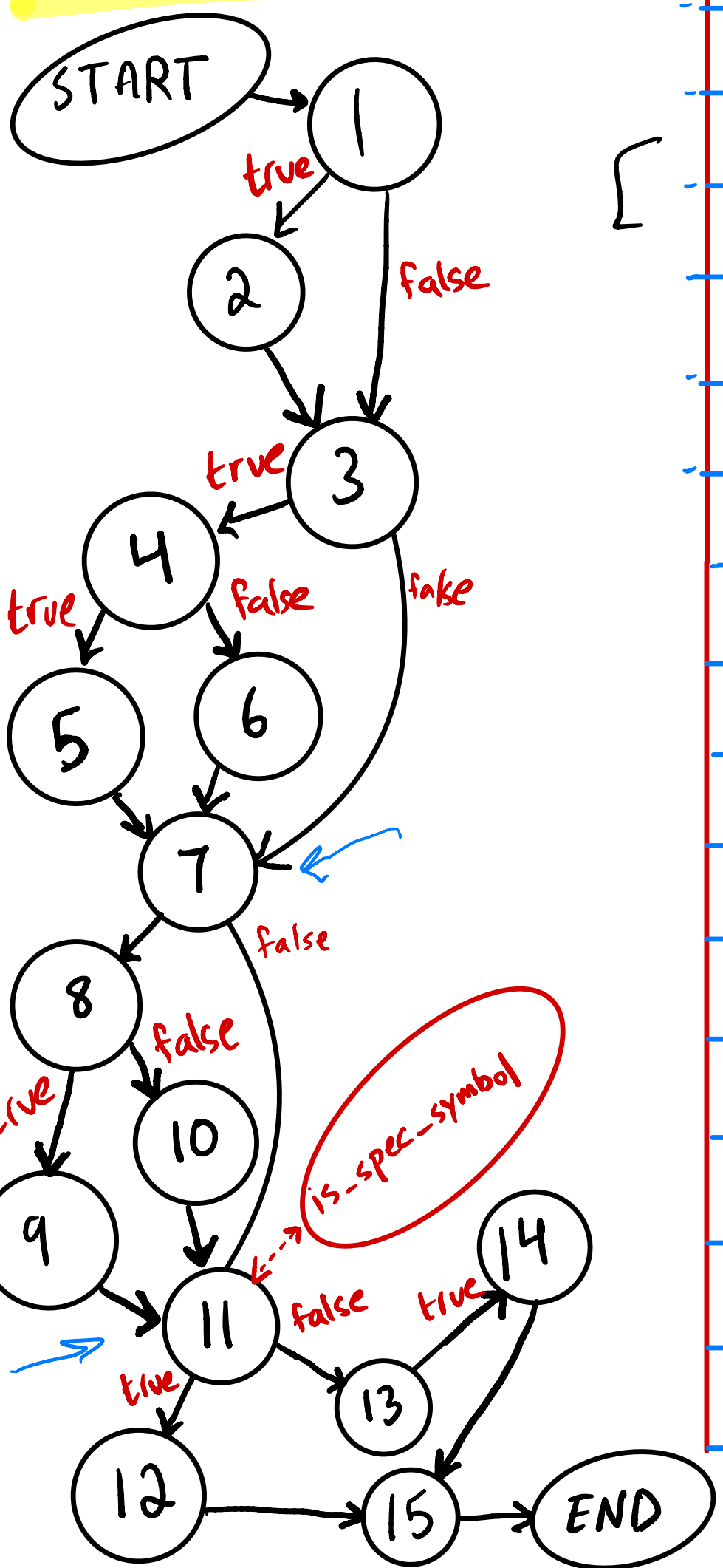
```
1 usage
static boolean is_token_end(int str_com_id, int res)
{
 if(res==-1)return(true); /* is eof token? */
 char ch = (char)res;
 if(str_com_id==1)          /* is string token */
    { if(ch=='"' || ch=='\n' || ch == '\r' || ch=='\t')   /* for string until meet another " */
         return true;
      else
         return false;
    }

 if(str_com_id==2)    /* is comment token */
   { if(ch=='\n' || ch == '\r' || ch=='\t')     /* for comment until meet end of line */
        return true;
      else
        return false;
   }

 if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
 if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59) return true;
                              /* others until meet blank or tab or 59 */
 return false;               /* other case,return FALSE */
}
```

get_char



Block Number	Lines	Entry	Exit	Function Call
1	45, 47, 48	45	48	
2	52	52	52	
3				
4				

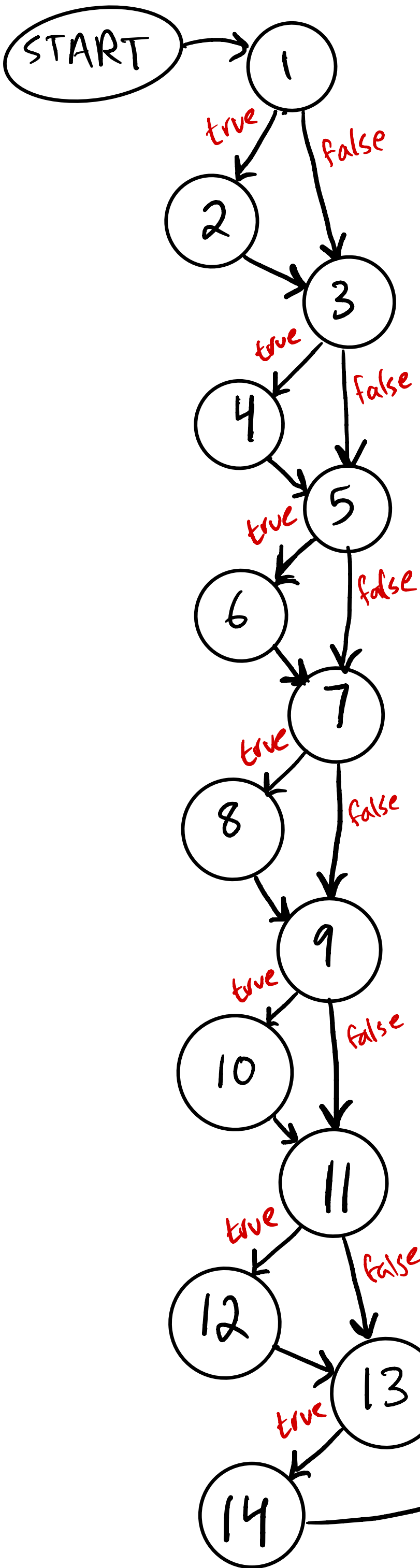
is_token_end



Block Number	Lines	Entry	Exit	Function Call
1	174a	174a	174a	
2	174b	174b	174b	
3	175, 176	175	176	
4	177	177	177	
5	178	178	178	
6	180	180	180	
7	183	183	183	
8	184	184	184	
9	185	185	185	
10	187	187	187	
11	190a	190a	190a	
12	190b	190b	190b	
13	191a	191a	191a	
14	191b	191b	191b	
15	193	193	193	

```
4 usages
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 == ',')
    {
        return true;
    }
    return false;     /* others return FALSE */
}
```

is_spec_symbol

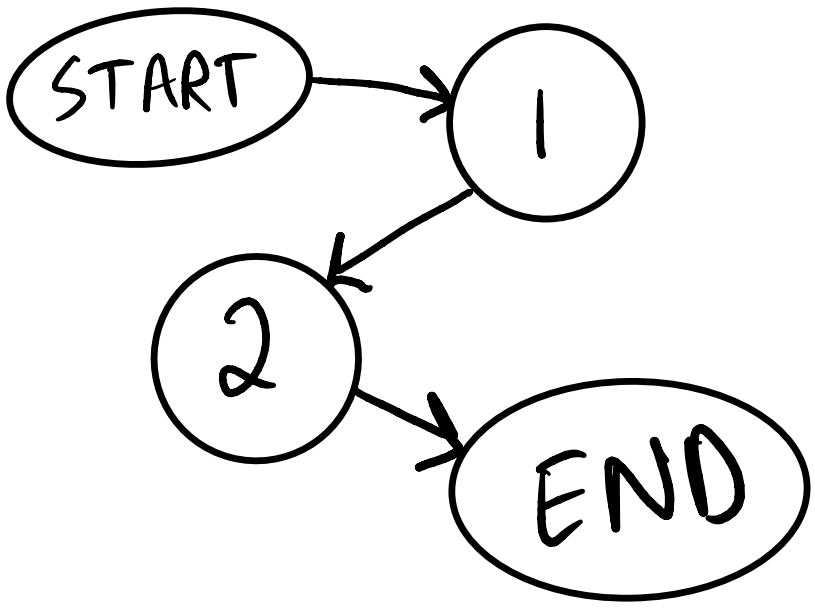


Block Number	Lines	Entry	Exit	Function Call
1	427	427	427	
2	429	429	429	
3	431	431	431	
4	433	433	433	
5	435	435	455	
6	437	437	437	
7	439	439	439	
8	441	441	441	
9	443	443	443	
10	445	445	445	
11	447	447	447	
12	449	449	449	
13	451	451	451	
14	453	453	453	
15	455	455	455	

```
4 usages
61  @   char unget_char (int ch,BufferedReader br) {
62        try {
63          br.reset();
64      } catch (IOException e) {
65          e.printStackTrace();
66      }
67           return 0;
68      }
69
70      /*****************************************************/
```

```
1 usage
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");
   }
 if(type==comment)
   {
    System.out.print("comment,\"" + tok + "\".\n");
   }
```

unget_char



Block Number	Lines	Entry	Exit	Function Call
1	63	63	63	
2	67	67	67	

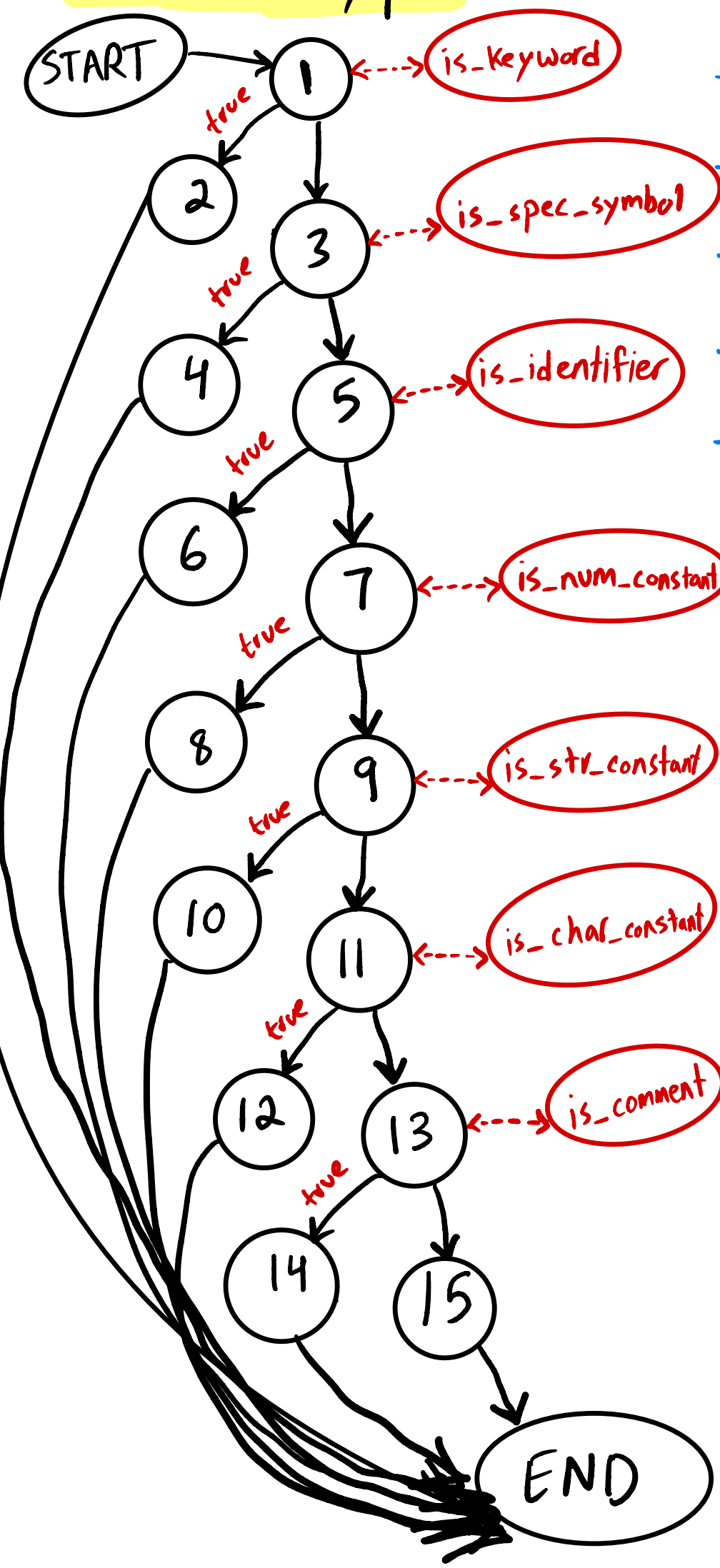
printf_token

Block Number	Lines	Entry	Exit	Function Call
1	220	220	220	
2	221	221	221	token_type
3	222	222	222	
4	224	224	224	
5	227	227	227	
6	229	229	229	
7	232a	232a	232a	
8	232b	232b	232b	print_spec_symbol
9	233	233	233	
10	235	235	235	
11	237	237	237	
12	239	239	239	
13	241	239	241	
14	243	243	243	
15	245	245	245	
16	247	247	247	
17	249	249	249	
18	251	251	251	

```
1 usage
203     static int token_type(String tok)
204     {
205      if(is_keyword(tok))return(keyword);
206      if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
207      if(is_identifier(tok))return(identifier);
208      if(is_num_constant(tok))return(num_constant);
209      if(is_str_constant(tok))return(str_constant);
210      if(is_char_constant(tok))return(char_constant);
211      if(is_comment(tok))return(comment);
212      return(error);                    /* else look as error token */
213     }
214
```

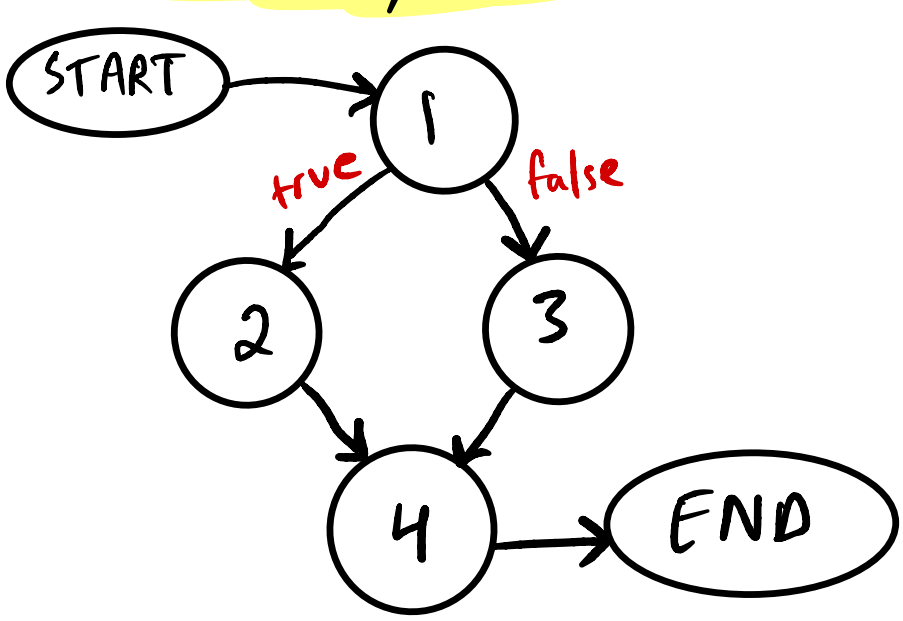
```
1 usage
276 @   static boolean is_keyword(String str)
277     {
278      if (str.equals("and") || str.equals("or") || str.equals("if") ||
279              str.equals("xor")||str.equals("lambda")||str.equals("=>"))
280          return true;
281       else
282          return false;
283     }
```

token_type



Block Number	Lines	Entry	Exit	Function Call
1	205a	205a	205a	is_keyword
2	205b	205b	205b	
3	206a	206a	206a	is_spec_symbol
4	206b	206b	206b	
5	207a	207a	207a	is_identifier
6	207b	207b	207b	
7	208a	208a	208a	is_num_constant
8	208b	208b	208b	
9	209a	209a	209a	is_str_constant
10	209b	209b	209b	
11	210a	210a	210a	is_char_constant
12	210b	210b	210b	
13	211a	211a	211a	is_comment
14	211b	211b	211b	
15	212	212	212	

is_keyword

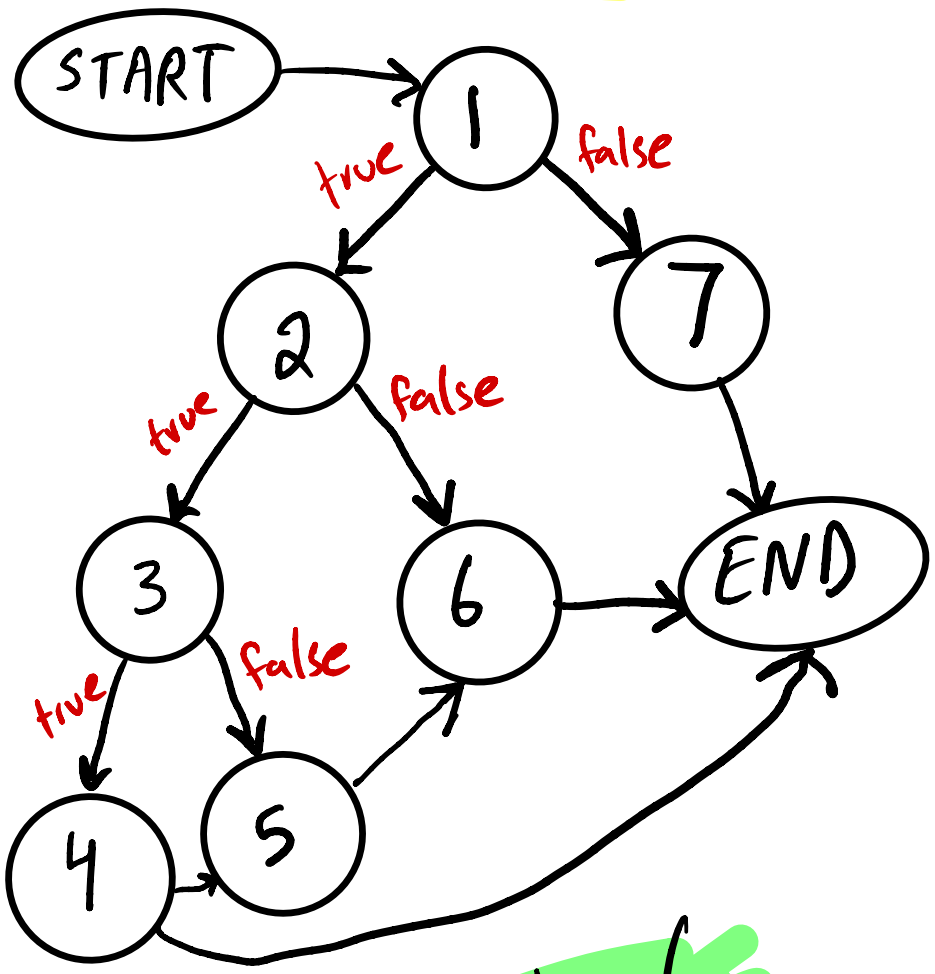


Block Number	Lines	Entry	Exit	Function Call
1	278	278	278	
2	280	280	280	
3	282	282	282	

```
1 usage
static boolean is_identifier(String str)
{
  int i=1;

  if ( Character.isLetter(str.charAt(0)) )
     {
      while(i < str.length() && str.charAt(i) !='\0' )   /* unti meet the end token sign */
          {
           if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
              i++;
           else
              return false;
          }      /* end WHILE */
       return false;
     }
  else
    return true;
}
```

```
1 usage
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' )
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     }
```

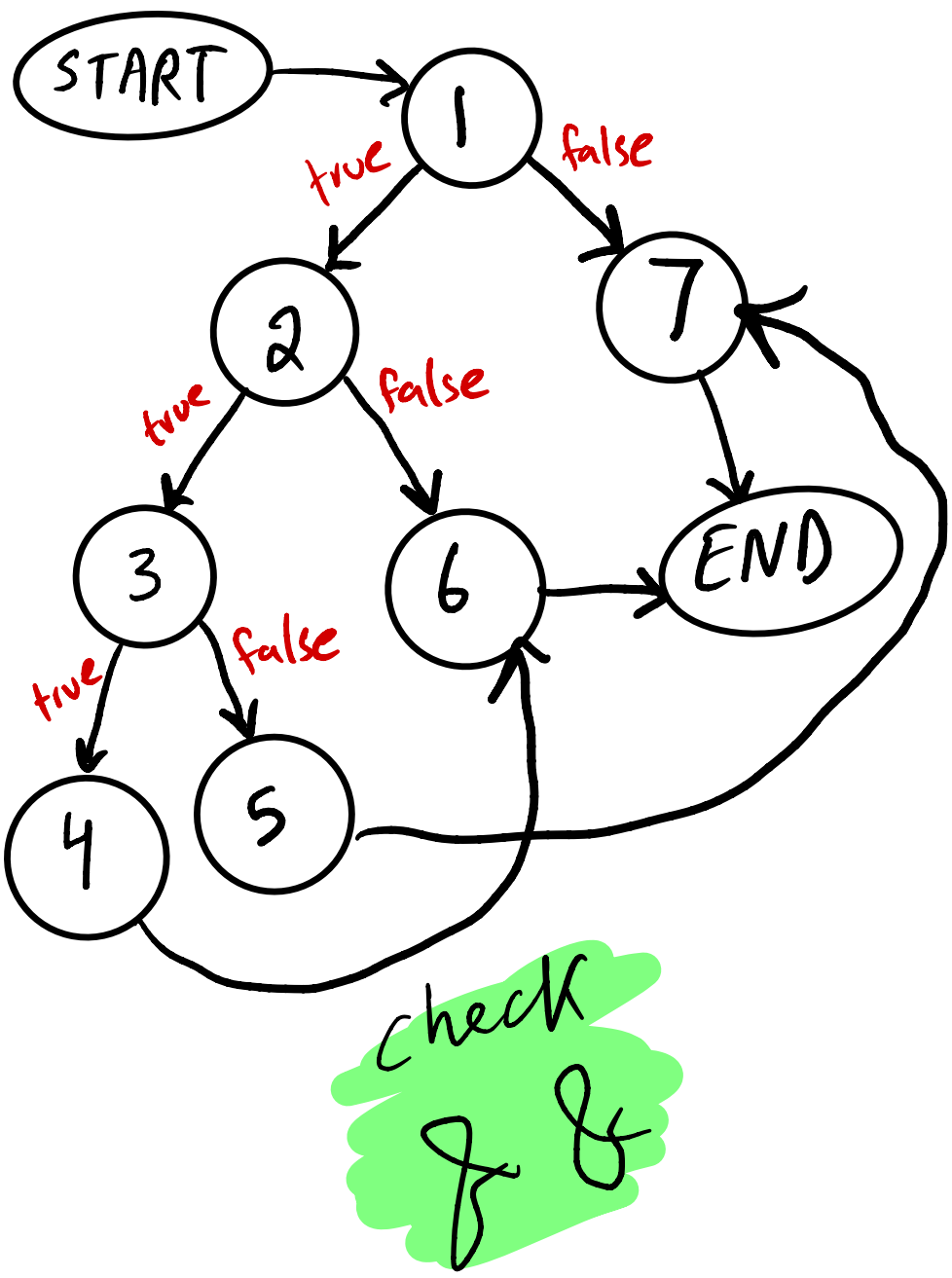
is_identifier



Block Number	Lines	Entry	Exit	Function Call
1	351, 353	351	353	
2	355	355	355	
3	357	357	357	
4	358	358	358	
5	360	360	360	
6	362	362	362	
7	365	365	365	
8				
9				

---

is_num_const



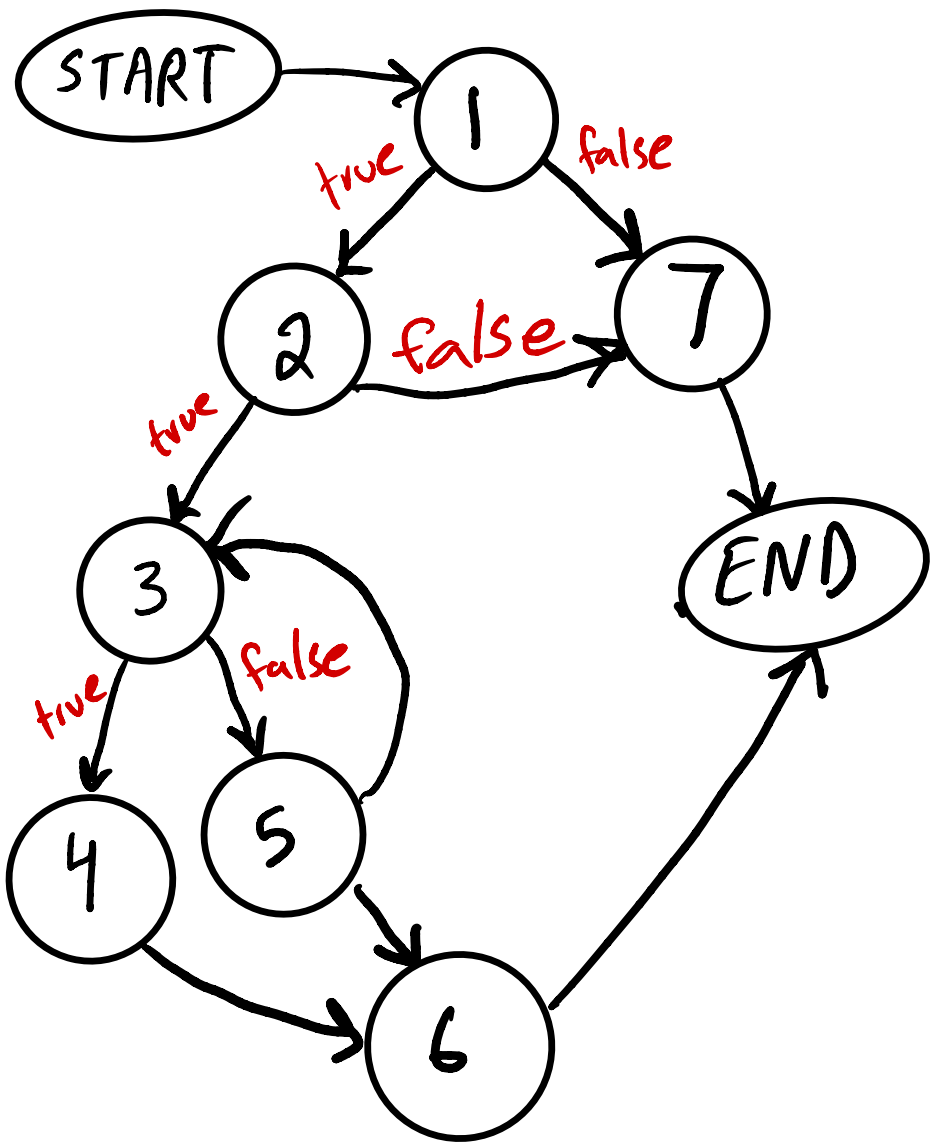
Block Number	Lines	Entry	Exit	Function Call
1	305, 307	305	307	
2	309	309	309	
3	311	311	311	
4	312	312	312	
5	314	314	314	
6	316	316	316	
7	319	319	319	
8				
9				

```
1 usage
static boolean is_str_constant(String str)
{
  int i=1;

  if ( str.charAt(0) =='"')
     { while (i < str.length() && str.charAt(i)!='\0')
         { if(str.charAt(i)=='"' )
             return true;        /* meet the second '"'          */
           else
           i++;
         }              /* end WHILE */
     return true;
     }
  else
    return false;       /* other return FALSE */
}
```

```
1 usage
static boolean is_char_constant(String str)
{
  if (str.length() > 2 || str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
     return true;
  else
     return false;
}
```

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

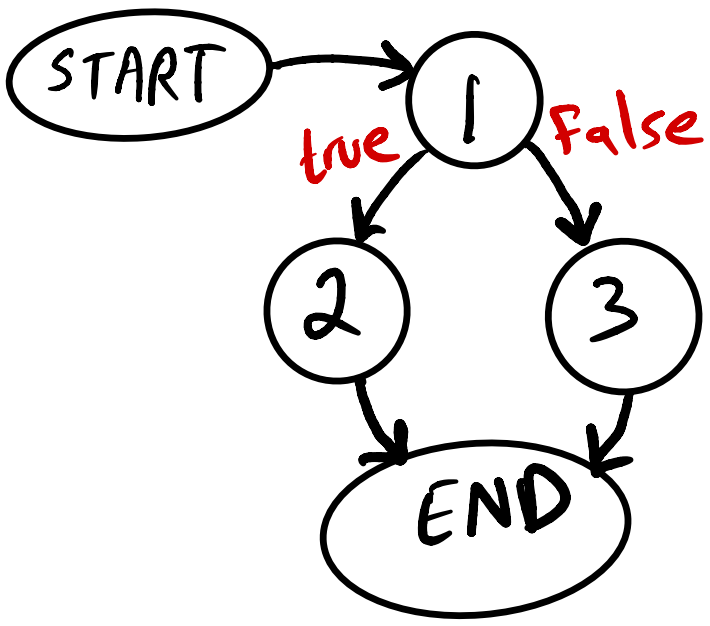
is_str_constant



Block Number	Lines	Entry	Exit	Function Call
1	329, 331	329	331	
2	332	332	332	
3	333	333	333	
4	334	334	334	
5	336	336	336	
6	338	338	338	
7	341	341	341	
8				
9				

---

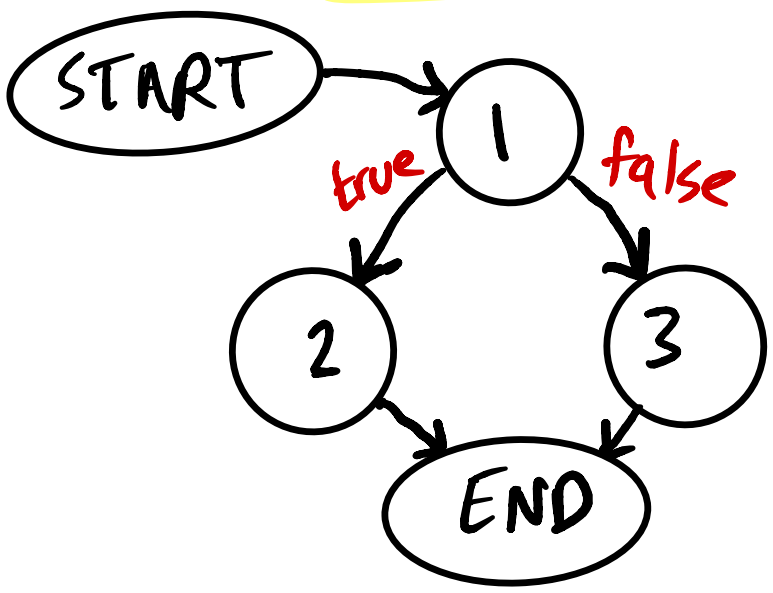
is_char_constant



Block Number	Lines	Entry	Exit	Function Call
1	292a, 292b	292a	292b	
2	293	293	293	
3	295	295	295	
4				
5				

---

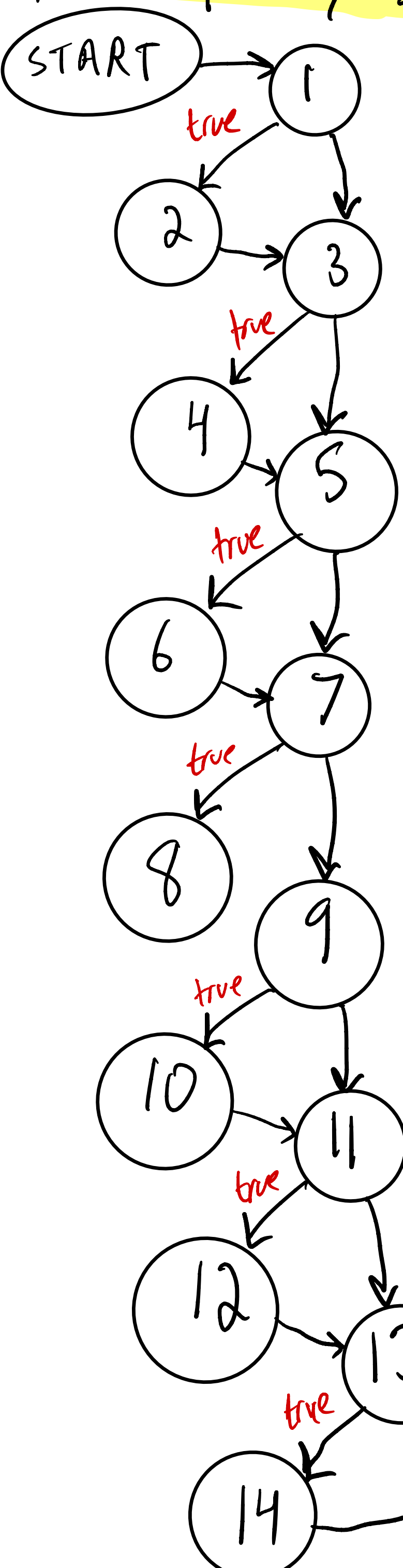
is_comment



Block Number	Lines	Entry	Exit	Function Call
1	265	265	265	
2	266	266	266	
3	268	268	268	

```
376 @       static void print_spec_symbol(String str)
377         {
378             if      (str.equals(")"))
379             {
380
381                     System.out.print("lparen.\n");
382                     return;
383             }
384             if (str.equals(")"))
385             {
386
387                     System.out.print("rparen.\n");
388                     return;
389             }
390             if (str.equals("["))
391             {
392                     System.out.print("lsquare.\n");
393                     return;
394             }
395             if (str.equals("]"))
396             {
397
398                     System.out.print("rsquare.\n");
399                     return;
400             }
401             if (str.equals("'"))
402             {
403                     System.out.print("quote.\n");
404                     return;
405             }
406             if (str.equals("`"))
407             {
408
409                     System.out.print("bquote.\n");
410                     return;
411             }
412
413             if (str.equals(","))
414             {
415                     System.out.print("comma.\n");
416                     return;
417             }
418         }
```

print_spec_symbol



Block Number	Lines	Entry	Exit	Function Call
1	378	378	378	
2	381,382	381	382	
3	384	384	384	
4	387,388	387	388	
5	390	390	390	
6	392,393	392	393	
7	395	395	395	
8	398,399	398	399	
9	401	401	401	
10	403,404	403	404	
11	406	406	406	
12	409,410	409	410	
13	413	413	413	
14	415,416	415	416	
15				