CSE-4321- 900
Software Testing and Maintenance Project
Faults Detected and Corrections

1. is_num_constant:
   Original code-
   ```
   while ( i <= str.length() && str.charAt(i) != '\0' )
   ```
   Modified code-
   ```
   while ( i < str.length() && str.charAt(i) != '\0' )
   ```
   Reason- The iterator 'i' cannot be equal to the length of the string as it can only go from 0 to n-1. The original code would have cause index out of range.

2. is_num_constant:
   Original code-
   ```
   if(Character.isDigit(str.charAt(i+1))) )
   ```
   Modified code-
   ```
   if(Character.isDigit(str.charAt(i))) )
   ```
   Reason- The iterator 'i' cannot be equal to the length of the string as it can only go from 0 to n-1. The original code would cause index out of range because when 'i' is n-1, i+1 would be equal to n which is incorrect.

3. is_str_constant:
   Original code- line 335
   ```
   return true;
   ```
   Modified code-
   ```
   return false;
   ```
   Reason- The while loop above this checks if there is a matching quote at the end for the starting quote. If the ending quote is not found and the loop is over, then it is not a string constant and therefore it should return false.

4. is_str_constant:
   Original code- line329
   ```
   while (i < str.length() && str.charAt(0)!='\0')
   ```
   Modified code-
   ```
   while (i < str.length() && str.charAt(i)!='\0')
   ```
   Reason- The original code only checks for the '\0' at the first position. But we need to check for the '\0' for each position so if we encounter it, we can end the loop.

5. is_identifier:
   Original code- line 359

```
return false;
```

Modified code-

```
return true;
```

Reason- When the while loop above the return ends without any problems, this shows that the character identifier.is

6. is_identifier:
   Original code- line 362
   ```
   return true;
   ```
   Modified code-
   ```
   return false;
   ```
   Reason- If the first character is not a letter, then it is not an identifier.

7. main:
   Original code- line 464
   System.exit(0);
   Modified code-
   return;
   Reason- The main function was not working with Junit with the exit statement. By replacing exit with return, I was able to run the function and keep the same functionality.

8. main:
   Original code- line 474,
   System.exit(0);
   Modified code-
   Removed System.exit(0);
   Reason- The main function was not working with Junit with the exit statement. By removing exit, I was able to run the function and keep the same functionality.

9. get_token:
   Original code- line 139,
   ```
   if(res == -1)
   ```
   Modified code-
   ```
   if(res == -1 && id != 2)
   ```
   Reason- Without modifying to id != 2, almost all the values were going inside this statements even if it was wrong. To fix this, I had to have another condition, so it goes in the block only for a specific block. The comments suggested that it was for EOF characters. The reason for choosing id != 2 was that all the other characters would mean that the token was over. However, if it was an ending quote, I still had to check if the start and end quote matched.

10. get_token:
Original code- line 148,

```
if(id == 1)
```

Modified code-

```
if(id == 2)
```

Reason- The comment next to this condition suggested that it checks for an ending quote. The value of 1 for id is for comments, and 2 is for quotes.

11. is_token_end:
Original code- line 176 and 183

```
176: if(str_com_id==1)
183: if(str_com_id==2)
```

Modified code-

```
176: if(str_com_id==2)
183: if(str_com_id==1)
```

Reason- The comments next to the lines 176 and 183 suggest that the statements are for string token and comments respectively. However, the str_com_id does not match with what the comments say. To correct this, I switched the two numbers so that they match.

12. print_token:
Modified code- line 249 to 251 added. This part of the code allows us to print values that are of string constant.

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

Reason- The reason for adding this was when I was testing a string constant value like "hello world", it was not giving me any output.

13. is_char_constant:
Original code- line 288,

```
if (str.length() > 2 && str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
```

Modified code-

```
if (str.length() == 2 && str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
```

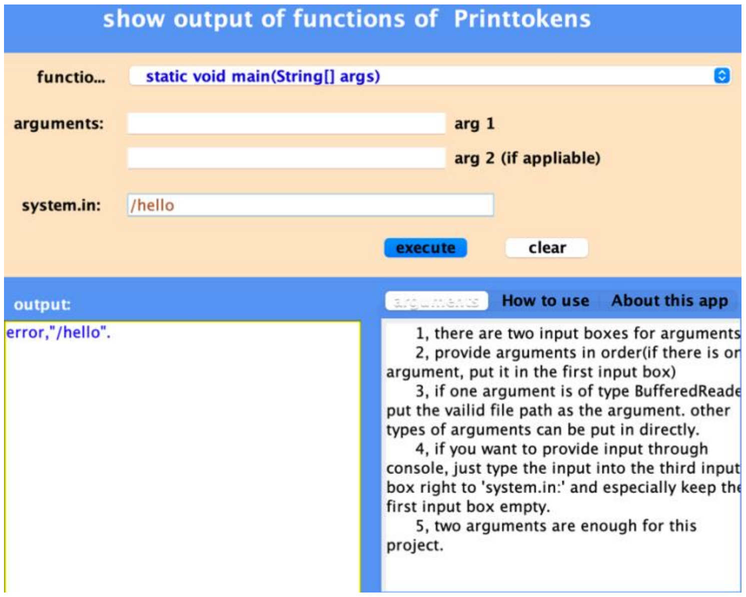
Reason- A character constant has to have a length of 2. The above code took characters that had more than 2 values which was wrong.

14. print_spec_symbol:
Original code-

```
if (str.equals("{"))
```
Modified code-
```
if (str.equals("("))
```
Reason- { is not a special symbol. Whereas (is.

15. is_spec_symbol:
Modified code-
```
if (c == '\'') //added
{
        return true;
 }
```
Reason- Missing if statement to check if ‘ is a special symbol.

16. is_spec_symbol:
Modified code-
```
if (c == '/') {
    return false;//return true;
}
```
Reason- changed return from true to false. I made this adjustment because the GUI did not consider ‘/’ as a special character. To make this consider as a part of the characters after it, I had to consider it as not a special character.



static void main(String[] args)

system.in: /hello

```
error,"/hello".
```

1, there are two input boxes for arguments
2, provide arguments in order(if there is or argument, put it in the first input box)
3, if one argument is of type BufferedReade put the vaild file path as the argument. other types of arguments can be put in directly.
4, if you want to provide input through console, just type the input into the third input box right to 'system.in:' and especially keep the first input box empty.
5, two arguments are enough for this project.