Test Cases

Team Members: Jayanth Reddy Gaddam(1002123569)

Jorge Catano (1002149092)

- Each test case must start with the "Start" node (0th node) and conclude with the "End" node, as shown in the CFG diagram in "CFG.pdf"
- "\0" is used to represent null strings
- The format for program-level testing simplifies test case representation as follows:

  [Start, 1, 2, non-mainMethod.7, 3, End]

- Here, non-mainMethod.7 represents the nodes from the 7th test case of the function non-mainMethod, while all other nodes belong to the main function.
- For example, consider test case #3 in main method:

  [Start, 1, 2, 6, 7, get_token.8, 8, 9, print_token.1, 10, get_token.1, 8, 11, End]

- In this representation, get_token.8 and get_token.1 refers respectively as:

  [Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 18, 20, 17,is_token_end, 21, 22, End]

  [Start, 1, 2, End]

- Replacing get_token.8 and get_tooken.1 with its expanded sequence gives:

[Start, 1, 2, 6, 7, [Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 18, 20, 17,is_token_end, 21, 22, End], 8, 9, print_token.1, 10, [Start, 1, 2, End], 8, 11, End]

- This process is repeated for all non-main method placeholders (get_token, print_token, etc.), expanding each according to its corresponding test case.
- We also followed the same approach for a few non main method functions to get end to end coverage.

- The purpose of this approach is to simplify the representation of system-wide test cases, avoiding unwieldy and overly complex test sequences.

1. String get_token(BufferedReader br)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 2, End]	\0 →null
2	[Start, 1, 3, 4, 5, 4, 6, 7, End]	\n\0 → null
3	[Start, 1, 3, 4, 6, 8, is_spec_symbol.1, 9, End]	(→ "("
4	[Start, 1, 3, 4, 6, 8, 10, 12, 14.2, 15, End]	\nx → "x"
5	[Start, 1, 3, 4, 6, 8, 10, 12, 13, 14, 15, End]	; → “;”
6	[Start, 1, 3, 4, 6, 8, 10, 11, 12, 14, 15, End]	“ → ""”"
7	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 18,19,21,22, End]	test → "test"
8	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 18, 20, 17,is_token_end, 21, 22, End]	b2 →"b2"
9	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 21, 23,is_spec_symbol.4, 24, End]	a]→ "a"
10	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 21, 23, 25, 26, End]	;note →";note"
11	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 21, 23, 25, 27, 28, End]	a’ →”a”
12	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 21, 23, 25, 27, 29,31, End]	test; → "test""
13	[Start, 1, 3, 4, 6, 8, 10, 12, 14, 16, 17, 21, 23, 25, 27, 29, 30, End]	“sample\” → ""sample\”"

2. boolean is_token_end(int str_com_id, int res)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 2, End]	0, -1 → true
2	[Start, 1, 3, 4, 5, End]	1, ‘ “ ‘→ true
3	[Start, 1, 3, 4, 6, End]	1, ‘a’ → false
4	[Start, 1, 3, 7, 8, 9, End]	2, '\n'→ true
5	[Start, 1, 3, 7, 8, 10, End]	2, ‘x’ → false
6	[Start, 1, 3, 7, 11, is_spec_symbol.1, 12, End]	0, ‘(‘ → true
7	[Start, 1, 3, 7, 11, 13, 14, End]	0, ‘ ‘ → true
8	[Start, 1, 3, 7, 11, 13, 15, End]	0, ‘a’ → false

3. boolean is_keyword(String str)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 2, End]	"xor" → true
2	[Start, 1, 3, End]	"hello" → false

4. boolean is_num_constant(String str)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 7, End]	"x2" → false
2	[Start, 1, 2, 6, End]	"4" → true
3	[Start, 1, 2, 3, 5, End]	“3.14” → false
4	[Start, 1, 2, 3, 4, 2, 6, End]	"123" → true

5	[Start, 1, 2, 3, 4, 2, 3,5, End]	“123.1” →false

5. boolean is_str_constant(String str)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 7, End]	1 → false
2	[Start, 1, 2, 6, End]	"\0" → false
3	[Start, 1, 2, 3, 4, End]	"" → true
4	[Start, 1, 2, 3, 5, 2, 6, End]	"Unclosed → false
5	[Start, 1, 2, 3, 5, 2, 3,4, End]	“hello” → true

6. boolean is_char_constant(String str)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 2, End]	";comment" → true
2	[Start, 1, 3, End]	"comment" → false

7. boolean is_comment(String ident)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 2, End]	"#a" → true
2	[Start, 1, 3, End]	"#ab" → false

8. boolean is_identifier(String str)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 7, End]	"2invalid" → false
2	[Start, 1, 2, 6, End]	"a" → true
3	[Start, 1, 2, 3, 5, End]	“a)” → false
4	[Start, 1, 2, 3, 4, 2, 6, End]	"x1y2z3" → true
5	[Start, 1, 2, 3, 4, 2, 3,5, End]	“no-hyphens” →false

9. boolean is_spec_symbol(char c)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 2, End]	**“(“**
2	[Start, 1, 3, 4, End]	")" → true
3	[Start, 1, 3,5, 6, End]	"[" → true
4	[Start, 1, 3,5, 7,8, End]	"]" → false
5	[Start, 1, 3, 5,7,9,10, End]	"/" → true
6	[Start, 1, 3, 5,7,9,11,12, End]	"’" → true
7	[Start, 1, 3, 5,7,9,11,13,14, End]	"," → true
8	[Start, 1, 3, 5,7,9,11,13,15, End]	";" → false

10. String token_type(String token)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1,is_keyword.1, 2, End]	"xor" → Printtokens.keyword
2	[Start, 1, 3,is_spec_symbol.3,4, End]	"[" → Printtokens.spec_symbol
3	[Start, 1, 3,5,is_identifier.4,6, End]	"myVar" → Printtokens.identifier
4	[Start, 1, 3,5,7,is_num_constant.4,8, End]	"42" → Printtokens.num_constant
5	[Start, 1, 3,5,7,9,is_str_constant.5,10, End]	""test"" → Printtokens.str_constant
6	[Start, 1, 3,5,7,9,11,is_char_constant.1,12, End]	"#x" → Printtokens.char_constant
7	[Start, 1, 3,5,7,9,11,13,is_comment.1,14, End]	";comment" → Printtokens.comment
8	[Start, 1, 3,5,7,9,11,13,15, End]	"@invalid" → Printtokens.error

11. void main(String[] args)

#	Test Path	Test Data/Input and Expected Output
1	[Start, 1, 3, 5, End]	Test Case #1: String[] args = {"input1.txt", "input2.txt"}. Multiple files provided. Expected Output: Error!, please give the token stream.
2	[Start, 1, 3, 4, 6, 7, 8, 11, End]	Test Case #2: String[] args = {"emptyFile.txt"}. Empty file input. Expected Output: (no output, file empty)

3	[Start, 1, 2, 6, 7, get_token.8, 8, 9, print_token.1, 10, get_token.1, 8, 11, End]	Test Case #3: String[] args = {"testInput.txt"} Input: xor 42 #z "test string" ;This is a comment Expected Output: keyword,"xor". numeric,42. character,"z". string,"test string". comment,";This is a comment".
4	[Start, 1, 2, 6, 7, get_token.3, 8, 9, print_token.3, 10, get_token.1, 8, 11, End]	Test Case #4: String[] args = {"specialSymbols.txt"} Input: ABC(5) Expected Output: identifier,"ABC". lparen. numeric,5. rparen.
5	[Start, 1, 2, 6, 7, get_token.4, 8, 9, print_token.4, 10, get_token.1, 8, 11, End]	Test Case #5: String[] args = {"complexTokens.txt"} Input: if (x => 10) Expected Output: keyword,"if". lparen. identifier,"x". keyword,"=>". numeric,10. rparen.

6	[Start, 1, 2, 6, 7, get_token.5, 8, 9, print_token.8, 10, get_token.1, 8, 11, End]	Test Case #6: String[] args = {} Console Input: a ; "hello" ABC and 5 #c Expected Output: identifier,"a". comment,"; "hello" ABC and 5 #c".
7	[Start, 1, 2, 6, 7, get_token.6, 8, 9, print_token.1, 10, get_token.1, 8, 11, End]	Test Case #7: String[] args = {"mixedTokens.txt"} Input: lambda (x) => x / 2 Expected Output: keyword,"lambda". lparen. identifier,"x". rparen. keyword,"=>". identifier,"x". slash. numeric,2.
8	[Start, 1, 2, 6, 7, get_token.7, 8, 9, print_token.4, 10, get_token.1, 8, 11, End]	Test Case #8: String[] args = {"errorTokens.txt"} Input: "unclosed string Expected Output: error,""unclosed string"."