System Test Plan
Logic Based Testing

By: Abdullah Sakallah
Mohammed Zakiuddin

Table of Contents

1. System Under Test

Compilers is a free and open-source project accessible on GitHub. The project under test covers a compiler that parses the input and generates a syntax tree. The project was created by developing context-free grammar in order to generate parsing rules and perform lexical analysis on the language.

1.1. Details

Project source code: https://github.com/Niljas/Compiler-Design-Project

Number of the components for Compilers

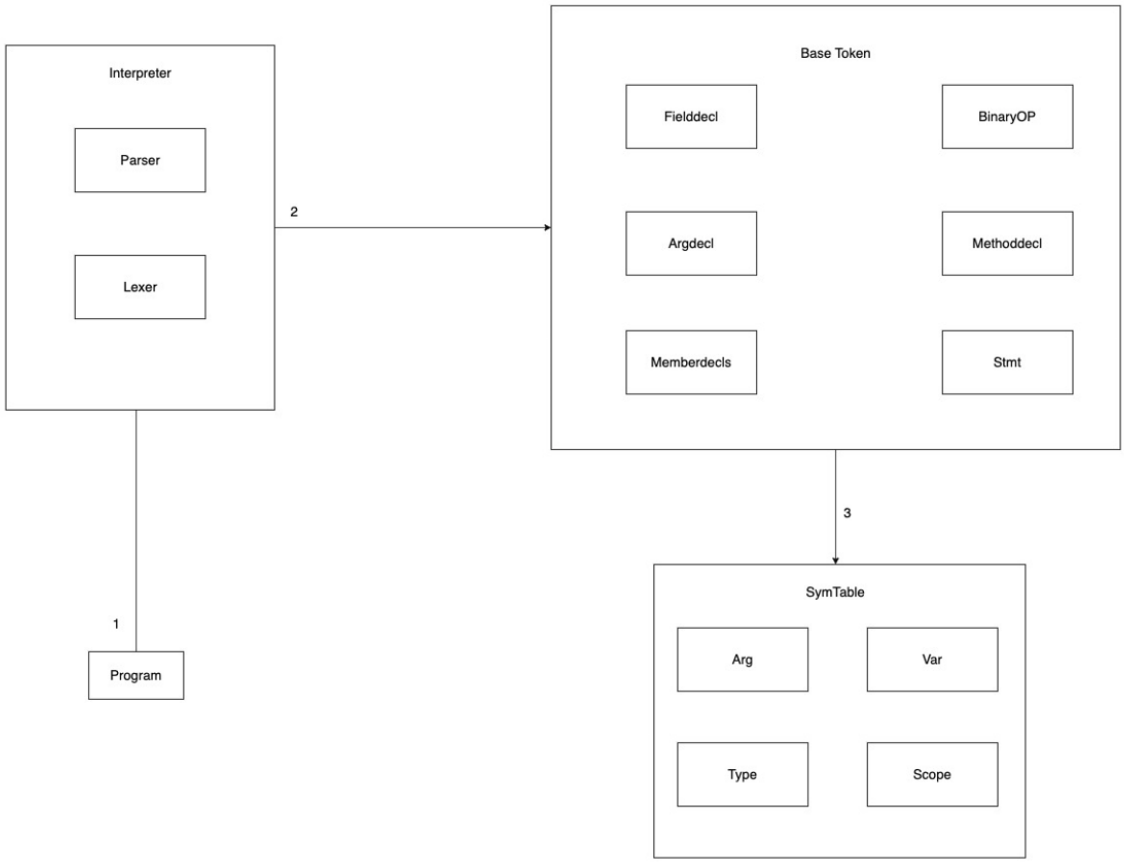
- Compiler Design Project (CDP):
  - 2 packages
  - 15 class files
  - 1 interface
  - 1 make file

Lines of code for each component

- Argdecl: 17 LOC
- BaseToken: 134 LOC
- BinaryOP: 19 LOC
- Checkable: 6 LOC (Interface)
- Compiler-Design-Project-Main: 12 LOC (Package 1)
- Expr: 207 LOC
- Fielddecl: 46 LOC
- Grammar: 97 LOC
- Interpreter: 37 LOC
- LexerRules: 17 LOC
- Makefile: 75 LOC (Make File)
- Memberdecls: 38 LOC
- Methoddecl: 66 LOC
- Name: 37 LOC
- ParserTest: 30 LOC
- Program: 16 LOC
- Stmt: 215 LOC
- SymTable: 74 LOC
- Tokens: 314 LOC (Package 2)
- TypeChecking: 32 LOC

Total LOC: 1489

1.2. Software Architecture



2. Test Environment

- The project under test is deployed under IntelliJ IDEA 2022.1.1 (Community Edition).
- It uses Junit 5 for unit testing.
- The Operating System that is used to run the tests is Windows 11 & Mac OS (Monterey 12.6).
- The version of Java that is being used is 17.0.2.

3. Logic Based Testing

3.1. BinaryOp – Public boolean isArithmetic()

Predicate [P1]: C1 || C2 || C3 || C4

[op.equals("+") || op.equals("-") || op.equals("*") || op.equals("/")]

Clauses:

- C1 = op.equals("+")
- C2 = op.equals("-")
- C3 = op.equals("*")
- C4 = op.equals("/")

Reachability:

- r(p1) = true (always reached)

P_{C1} determination =

P = C1 ⋁ C2 ⋁ C3 ⋁ C4

P_{C1} = P_{C1} = true ⊕ P_{C1} = false

= true ⋁ C2 ⋁ C3 ⋁ C4 ⊕ false ⋁ C2 ⋁ C3 ⋁ C4

= true ⊕ C2 ⋁ C3 ⋁ C4 – Using p⊕q = (p v q) ^ !(p^q)

= (true ⋁ (C2 ⋁ C3 ⋁ C4)) ⋀ ¬(true ⋀ (C2 ⋁ C3 ⋁ C4))

= true ⋀ ¬(C2 ⋁ C3 ⋁ C4)

= ¬(C2 ⋁ C3 ⋁ C4)

P_{C2} determination =

P = C1 ⋁ C2 ⋁ C3 ⋁ C4

P_{C2} = P_{C2} = true ⊕ P_{C2} = false

= true ⋁ C1 ⋁ C3 ⋁ C4 ⊕ false ⋁ C1 ⋁ C3 ⋁ C4

= true ⊕ C1 ⋁ C3 ⋁ C4 – Using p⊕q = (p v q) ^ !(p^q)

= (true ⋁ (C1 ⋁ C3 ⋁ C4)) ⋀ ¬(true ⋀ (C1 ⋁ C3 ⋁ C4))

= true ⋀ ¬(C1 ⋁ C3 ⋁ C4)

= ¬(C1 ⋁ C3 ⋁ C4)

P_{C3} determination =

P = C1 ⋁ C2 ⋁ C3 ⋁ C4

P_{C3} = P_{C3} = true ⊕ P_{C3} = false

= C1 ⋁ C2 ⋁ true ⋁ C4 ⊕ C1 ⋁ C2 ⋁ false ⋁ C4

= true ⊕ C1 ⋁ C2 ⋁ C4 – Using p⊕q = (p v q) ^ !(p^q)

= (true ⋁ (C1 ⋁ C2 ⋁ C4)) ⋀ ¬(true ⋀ (C1 ⋁ C2 ⋁ C4))

= true ⋀ ¬(C1 ⋁ C2 ⋁ C4)

= ¬(C1 ⋁ C2 ⋁ C4)

P_{C4} determination =

P = C1 ⋁ C2 ⋁ C3 ⋁ C4

P_{C4} = P_{C4} true ⊕ P_{C4} = false

= true ⋁ C1 ⋁ C2 ⋁ C3 ⊕ false ⋁ C1 ⋁ C2 ⋁ C3

= true ⊕ C1 ⋁ C2 ⋁ C3 – Using p⊕q = (p v q) ^ !(p^q)

= (true ⋁ (C1 ⋁ C2 ⋁ C3)) ⋀ ¬(true ⋀ (C1 ⋁ C2 ⋁ C3))

= true ⋀ ¬(C1 ⋁ C2 ⋁ C3)

= ¬(C1 ⋁ C2 ⋁ C3)

Predicate Table (PT) [Unique ID]:

Row ID	C_1	C_2	C_3	C_4	P1	P_{C1}	P_{C2}	P_{C3}	P_{C4}
1	T	T	T	T	T	F	F	F	F
2	T	T	T	F	T	F	F	F	F
3	T	T	F	T	T	F	F	F	F
4	T	T	F	F	T	F	F	F	F
5	T	F	T	T	T	F	F	F	F
6	T	F	T	F	T	F	F	F	F
7	T	F	F	T	T	F	F	F	F
8	T	F	F	F	T	T	F	F	F
9	F	T	T	T	T	F	F	F	F
10	F	T	T	F	T	F	F	F	F
11	F	T	F	T	T	F	F	F	F
12	F	T	F	F	T	F	T	F	F
13	F	F	T	T	T	F	F	F	F
14	F	F	T	F	T	F	F	T	F
15	F	F	F	T	T	F	F	F	T
16	F	F	F	F	F	T	T	T	T

3.2. Expr - public String toString()

Predicate[P1]: C1 || C2 case 0: case 12:

Predicate[P2]: C3 case 1:

Predicate[P3]: C4 case 2:

Predicate[P4]: C5 case 3:

Predicate[P5]: C6 case 4:

Predicate[P6]: C7 bool

Predicate[P7]: C8 case 5:

Predicate[P8]: C9 case 6:

Predicate[P9]: C10 case 7:

Predicate[P10]: C11 case 8:

Predicate[P11]: C12 case 9:

Predicate[P12]: C13 case 10:

Predicate[P13]: C14 ret.length > 0

Predicate[P14]: C15 case 11:

Clauses:

- C1 = this.typeNumber == 0 (case 0)
- C2 = this.typeNumber == 12 (case 12)
- C3 = this.typeNumber == 1 (case 1)
- C4 = this.typeNumber == 2 (case 2)
- C5 = this.typeNumber == 3 (case 3)
- C6 = this.typeNumber == 4 (case 4)
- C7 = bool
- C8 = this.typeNumber == 5 (case 5)
- C9 = this.typeNumber == 6 (case 6)
- C10 = this.typeNumber == 7 (case 7)
- C11 = this.typeNumber == 8 (case 8)
- C12 = this.typeNumber == 9 (case 9)
- C13 = this.typeNumber == 10 (case 10)
- C14 = ret.length > 0

- C15 = this.typeNumber == 11 (case 11)

Reachability:

r(p1) = true (always reached)

r(p2) = !p1 = (this.typeNumber != 0 && this.typeNumber != 12)

r(p3) = r(p2) && !p2 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1)

r(p4) = r(p3) && !p3 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1)
&&(this.typeNumber != 2)

r(p5) = r(p4) && !p4 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3)

r(p6) = r(p5) && p5 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber == 4)

r(p7) = r(p5) && !p5 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4)

r(p8) = r(p7) && !p7 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&
(this.typeNumber != 5)

r(p9) = r(p8) && !p8 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&
(this.typeNumber != 5) && (this.typeNumber != 6)

r(p10) = r(p9) && !p9 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&
(this.typeNumber != 5) && (this.typeNumber != 6) && (this.typeNumber != 7)

r(p11) = r(p10) && !p10 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&

(this.typeNumber != 5) && (this.typeNumber != 6) && (this.typeNumber != 7) &&
(this.typeNumber != 8)

r(p12) = r(p11) && !p11 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&
(this.typeNumber != 5) && (this.typeNumber != 6) && (this.typeNumber != 7) &&
(this.typeNumber != 8) && (this.typeNumber != 9)

r(p13) = r(p12) && p12 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&
(this.typeNumber != 5) && (this.typeNumber != 6) && (this.typeNumber != 7) &&
(this.typeNumber != 8) && (this.typeNumber != 9) && (this.typeNumber == 10)

r(p14) = r(p12) && !p12 =
(this.typeNumber != 0 && this.typeNumber != 12) && (this.typeNumber != 1) &&
(this.typeNumber != 2) && (this.typeNumber != 3) && (this.typeNumber != 4) &&
(this.typeNumber != 5) && (this.typeNumber != 6) && (this.typeNumber != 7) &&
(this.typeNumber != 8) && (this.typeNumber != 9) && (this.typeNumber != 10)

$P1_{C1}$ determination =

$P = C1 \parallel C2$

$P_{C1} = P_{C1}$ true $\oplus$ P_{C1} = false

P_{C1} = true $\vee$ C2 $\oplus$ false $\vee$ C2

P_{C1} = true $\oplus$ C2

P_{C1} = true $\oplus$ C2

$P_{C1} = \neg C2$

$P1_{C2}$ determination =

$P = C1 \parallel C2$

$P_{C2} = P_{C2}$ true $\oplus$ P_{C2} = false

P_{C2} = C1 $\vee$ true $\oplus$ C1 $\vee$ false

P_{C1} = true $\oplus$ C1

$P_{C1} = \neg C1$

Row ID	C1	C2	P	P_{C1}	P_{C2}
1	T	T	T	F	F
2	T	F	T	T	F
3	F	T	T	F	T
4	F	F	F	T	T

$P2_{C3}$ determination =

$P = C3$

$P2_{C3} = P2_{C3}$ = true $\oplus$ $P2_{C3}$ = false

$P2_{C3}$ = true $\oplus$ false

$P2_{C3}$ = true

Row ID	C3	P	P_{C3}
1	T	T	T
2	F	F	T

$P3_{C4}$ determination =

$P = C4$

$P_{C4} = P_{C4}$ true $\oplus$ P_{C4} = false

P_{C4} = true $\oplus$ false

P_{C4} = true

Row ID	C4	P	P_{C4}
1	T	T	T
2	F	F	T

$P4_{C5}$ determination =

$P = C5$

$P_{C5} = P_{C5}$ true $\oplus$ P_{C5} = false

P_{C5} = true $\oplus$ false

P_{C5} = true

Row ID	C5	P	P_{C5}
1	T	T	T
2	F	F	T

$P5_{C6}$ determination =

$P = C6$

$P_{C6} = P_{C6}$ true $\oplus$ P_{C6} = false

P_{C6} = true $\oplus$ false

P_{C6} = true

Row ID	C6	P	P_{C6}
1	T	T	T
2	F	F	T

$P6_{C7}$ determination =

$P = C7$

$P_{C7} = P_{C7}$ true $\oplus$ P_{C7} = false

P_{C7} = true $\oplus$ false

P_{C7} = true

Row ID	C7	P	P_{C7}
1	T	T	T
2	F	F	T

$P7_{C8}$ determination =

$P = C8$

$P_{C8} = P_{C8}$ true $\oplus$ P_{C8} = false

P_{C8} = true $\oplus$ false

P_{C8} = true

Row ID	C8	P	P_{C8}
1	T	T	T
2	F	F	T

$P8_{C9}$ determination =

$P = C9$

$P_{C9} = P_{C9}$ true $\oplus$ P_{C9} = false

P_{C9} = true $\oplus$ false

P_{C9} = true

Row ID	C9	P	P_{C9}
1	T	T	T
2	F	F	T

$P9_{C10}$ determination =

$P = C10$

$P_{C10} = P_{C10}$ true $\oplus$ P_{C10} = false

P_{C10} = true $\oplus$ false

P_{C10} = true

Row ID	C10	P	P_{C10}
1	T	T	T
2	F	F	T

$P10_{C11}$ determination =

$P = C11$

$P_{C11} = P_{C11}$ true $\oplus$ P_{C11} = false

P_{C11} = true $\oplus$ false

P_{C11} = true

Row ID	C11	P	P_{C11}
1	T	T	T
2	F	F	T

$P11_{C12}$ determination =

$P = C12$

$P_{C12} = P_{C12}$ true $\oplus$ P_{C12} = false

P_{C12} = true $\oplus$ false

P_{C12} = true

Row ID	C12	P	P_{C12}
1	T	T	T
2	F	F	T

$P12_{C13}$ determination =

$P = C13$

$P_{C13} = P_{C13}$ true $\oplus$ P_{C13} = false

P_{C13} = true $\oplus$ false

P_{C13} = true

Row ID	C13	P	P_{C13}
1	T	T	T
2	F	F	T

$P13_{C14}$ determination =

$P = C14$

$P_{C14} = P_{C14}$ true $\oplus$ P_{C14} = false

P_{C14} = true $\oplus$ false

P_{C14} = true

Row ID	C14	P	P_{C14}
1	T	T	T
2	F	F	T

$P14_{C15}$ determination =

$P = C15$

$P_{C15} = P_{C15}$ true $\oplus$ P_{C15} = false

P_{C15} = true $\oplus$ false

P_{C15} = true

Row ID	C15	P	P_{C15}
1	T	T	T
2	F	F	T

3.3. BinaryOp - public boolean isLogical()

Predicate [P1]: C1 || C2 [op.equals("||") || op.equals("&&")]

Clauses:

- C1 = op.equals("||")
- C2 = op.equals("&&")

Reachability Predicates:

r(P1): True (always reached)

P1 = C1 ⋁ C2

P_{C1} determination =

$P_{C1} = P_{C1}$ = true ⊕ P_{C1} = false

P_{C1} = true ⋁ C2 ⊕ false ⋁ C2

P_{C1} = true ⊕ C2

P_{C1} = ¬C2

P_{C2} determination =

$P_{C2} = P_{C2}$ = true ⊕ P_{C2} = false

P_{C2} = C1 ⋁ true ⊕ C1 ⋁ false

P_{C2} = true ⊕ C1

P_{C2} = ¬C1

Row ID	C1	C2	P	P_{C1}	P_{C2}
1	T	T	T	F	F
2	T	F	T	T	F
3	F	T	T	F	T
4	F	F	F	T	T

4. Test Requirements

4.1. CACC Coverage Criteria

MUT: [isArithmetic()]

TR	PT ID	Row	Coverage	Feasible?
1	P1	(8, 16)	P_{C1} where P_1 = True	Yes, op.equals("+") == true
2	P1	(8, 16)	P_{C1} where P_1 = False	Yes, doesn't equal any
3	P1	(12, 16)	P_{C2} where P_1 = True	Yes, op.equals("-") == true
4	P1	(12, 16)	P_{C2} where P_1 = False	Yes, doesn't equal any
5	P1	(14, 16)	P_{C3} where P_1 = True	Yes, op.equals("*") == true
6	P1	(14, 16)	P_{C3} where P_1 = False	Yes, doesn't equal any
7	P1	(15, 16)	P_{C4} where P_1 = False	Yes, op.equals("/") == true
8	P1	(15, 16)	P_{C4} where P_1 = False	Yes, doesn't equal any

MUT: toString()

TR	PT ID	Row	Coverage	Feasible?
1	P1	(2, 4)	P_{C1} where P_1 = True	Yes, this.typeNumber == 0
2	P1	(2, 4)	P_{C1} where P_1 = False	Yes, this.typeNumber != 0 && this.typeNumber != 12
3	P1	(3, 4)	P_{C2} where P_1 = True	Yes, this.typeNumber == 12
4	P1	(3, 4)	P_{C2} where P_1 = False	Yes, doesn't equal any
5	P2	(1, 2)	P_{C3} where P_2 = True	Yes, this.typeNumber == 1›
6	P2	(1, 2)	P_{C3} where P_2 = False	Yes, doesn't equal any
7	P3	(1, 2)	P_{C4} where P_3 = False	Yes, this.typeNumber == 2
8	P3	(1, 2)	P_{C4} where P_3 = False	Yes, doesn't equal any
9	P4	(1, 2)	P_{C5} where P_4 = True	Yes, this.typeNumber == 3
10	P4	(1, 2)	P_{C5} where P_4 = False	Yes, doesn't equal any
11	P5	(1, 2)	P_{C6} where P_5 = True	Yes, this.typeNumber == 4
12	P5	(1, 2)	P_{C6} where P_5 = False	Yes, doesn't equal any
13	P6	(1, 2)	P_{C7} where P_6 = True	Yes, bool == true
14	P6	(1, 2)	P_{C7} where P_6 = False	Yes, bool == false
15	P7	(1, 2)	P_{C8} where P_7 = False	Yes, this.typeNumber == 5
16	P7	(1, 2)	P_{C8} where P_7 = False	Yes, doesn't equal any
17	P8	(1, 2)	P_{C9} where P_8 = True	Yes, this.typeNumber == 6
18	P8	(1, 2)	P_{C9} where P_8 = False	Yes, doesn't equal any
19	P9	(1, 2)	P_{C10} where P_9 = True	Yes, this.typeNumber == 7
20	P9	(1, 2)	P_{C10} where P_9 = False	Yes, doesn't equal any
21	P10	(1, 2)	P_{C11} where P_{10} = True	Yes, this.typeNumber == 8
22	P10	(1, 2)	P_{C11} where P_{10} = False	Yes, doesn't equal any
23	P11	(1, 2)	P_{C12} where P_{11} = False	Yes, this.typeNumber == 9
24	P11	(1, 2)	P_{C12} where P_{11} = False	Yes, doesn't equal any
25	P12	(1, 2)	P_{C13} where P_{12} = True	Yes, this.typeNumber == 10

26	P12	(1, 2)	P_{C13} where P_{12} = False	Yes, doesn't equal any
27	P13	(1, 2)	P_{C14} where P_{13} = True	Yes, ret.length > 0
28	P13	(1, 2)	P_{C14} where P_{13} = False	Yes, ret.length <= 0
29	P14	(1, 2)	P_{C15} where P_{14} = True	Yes, this.typeNumber == 11
30	P14	(1, 2)	P_{C15} where P_{14} = False	Yes, doesn't equal any

MUT: isLogical()

TR	PT ID	Row	Coverage	Feasible?
1	P1	(2, 4)	P_{C1} where P_1 = True	Yes, op.equals("||") == true
2	P1	(2, 4)	P_{C1} where P_1 = False	Yes, doesn't equal any
3	P1	(3, 4)	P_{C2} where P_1 = True	Yes, op.equals("&&") == true
4	P1	(3, 4)	P_{C2} where P_1 = False	Yes, doesn't equal any

5. Test Results with Traceability

Test ID	Targeted TR	MUT	Input	Observed Output	Result
1	TR1	isArithmetic	+	True	Pass
2	TR2	isArithmetic	#	False	Pass
3	TR3	isArithmetic	-	True	Pass
4	TR4	isArithmetic	@	False	Pass
5	TR5	isArithmetic	*	True	Pass
6	TR6	isArithmetic	!	False	Pass
7	TR7	isArithmetic	/	True	Pass
8	TR8	isArithmetic	$	False	Pass
9	TR1	toString()	Case 0, “foo”	“foo”	Pass
10	TR2	toString()	Case 1, 1	“1”	Pass
11	TR3	toString()	Case 12, “foo”	“foo”	Pass
12	TR4	toString()	Case 2, 1	“1.0”	Pass
13	TR5	toString()	Case 1, 100	“100”	Pass
14	TR6	toString()	Case 2, 100	“100.0”	Pass
15	TR7	toString()	Case 2, 2	“2.0”	Pass
16	TR8	toString()	Case 3, “foo”	“foo”	Pass
17	TR9	toString()	Case 3,“return 0;”	“return 0;”	Pass
18	TR10	toString()	Case 0, “string”	“string”	Pass
19	TR11	toString()	Case 4, true	“true”	Pass
20	TR12	toString()	Case 5, 1	“1”	Pass
21	TR13	toString()	Case 4, bool = “true”	bool == true	Pass
22	TR14	toString()	Case 4, bool == “false”	bool == false	Pass
23	TR15	toString()	Case 5, 111	“111”	Pass
24	TR16	toString()	Case 6, “boo”,”foo”	(boo foo)	Pass
25	TR17	toString()	Case 6, “string s” “int x”	(string s int x)	Pass
26	TR18	toString()	Case 7, “int”, “x”	(int)x	Pass
27	TR19	toString()	Case 7, “int”, “y”	(int)y	Pass
28	TR20	toString()	Case 8, “y”	(y + y)	Pass
29	TR21	toString()	Case 8, “1”	(1 + 1)	Pass
30	TR22	toString()	Case 9, “1”	(1 ? 1 : 1)	Pass
31	TR23	toString()	Case 9, “len”	(len ? len : len)	Pass
32	TR24	toString()	Case 10, “foo”, “x”, “len”	foo(x, len)	Pass
33	TR25	toString()	Case 10, “5”, “1”, “sum”	sum(5, 1)	Pass
34	TR26	toString()	Case 11, true, “sum”	sum()	Pass
35	TR27	toString()	Case 10, “x” , “y”	(y, x)	Pass
36	TR28	toString()	Case 10, “”	()	Pass
37	TR29	toString()	Case 11, false “sum”	sum()	Pass
38	TR30	toString()	Case default, typeNumber=100	“”	Pass
39	TR1	isLogical()	“||”	true	Pass

40	TR2	isLogical()	“!”	false	Pass
41	TR3	isLogical()	“&&”	true	Pass
42	TR4	isLogical()	“~”	false	Pass