STT SUMMER 2022

REPORT ON CODE COVERAGE USING AUTOTESTER

STUDENT DETAILS

Student Name: Syed Ahtsham

ASSIGNMENT DETAILS

Course: Software Testing Techniques - STT

Submitted To: Dr. Mudassar Azam Sindhu

Testing the TriangleTest Code using AutoTester

Software used: Autotester

Acronyms Table:

Sr. No.	Acronym	Definition
1.	SCC/NCC	Statement Coverage Criteria/Node Coverage Criteria
2.	DCC/PCC	Decision Coverage Criteria/Predicate Coverage Criteria
3.	CCC/CCC	Condition Coverage Criteria/Clause Coverage Criteria
4.	CDCC/ECC	Condition Decision Coverage Criteria/Edge Coverage Criteria
5.	MCDC/RACC	Multi Condition Decision Coverage Criteria/Restricted Active Clause Coverage

Table 1.1

1. Node Coverage Criteria or Statement Coverage Criteria (NCC/SCC):

Autotester supports the NC. Test cases generated by the autotester are not minimized.

Control Flow Graph (CFG)	Code under Test	Test Paths generated by Autotester

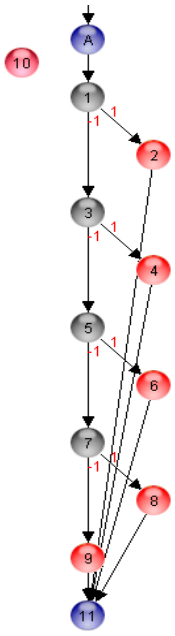
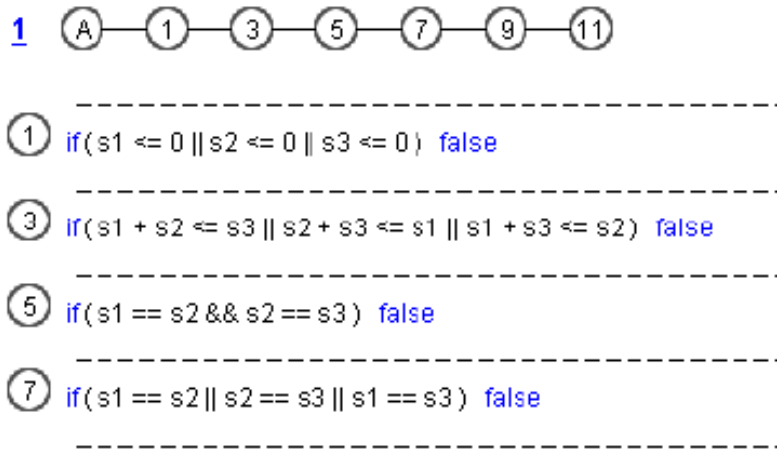
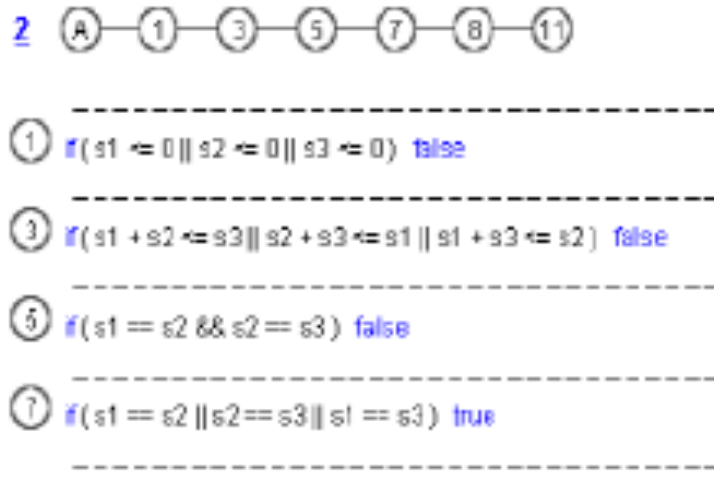
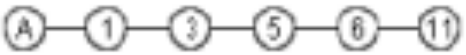
	Ⓐ public static Type TriangleTest (int s1 , int s2 , int s3) {
	① if (s1 <= 0 || s2 <= 0 || s3 <= 0) {
	② return Type . BADSIDE ;
	}
	③ else {
	if (s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) {
	④ return Type . NOTRIANGLE ;
	}
	⑤ else {
	if (s1 == s2 && s2 == s3) {
	⑥ return Type . EQUILATERAL ;
	}
	⑦ else {
	if (s1 == s2 || s2 == s3 || s1 == s3) {
	⑧ return Type . ISOSCELES ;
	}
	⑨ else {
	return Type . SCALENE ;
	}
	⑩ }
	}
	}
	}
	⑪ // end
		1 Ⓐ—①—③—⑤—⑦—⑨—⑪
		2 Ⓐ—①—③—⑤—⑦—⑧—⑪
		3 Ⓐ—①—③—⑤—⑥—⑪
		4 Ⓐ—①—③—④—⑪
		5 Ⓐ—①—②—⑪

Fig: Table 1

a) Test Cases for execution of Test Paths for Node Coverage:





3 

① if(s1 <= 0 || s2 <= 0 || s3 <= 0) false

③ if(s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) false

⑤ if(s1 == s2 && s2 == s3) true

4 

① if(s1 <= 0 || s2 <= 0 || s3 <= 0) false

③ if(s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) true

5 

① if(s1 <= 0 || s2 <= 0 || s3 <= 0) true

Fig: Test Cases Table

Test Path	Test Case	Expected Output	Observed Output	Verdict
1.	<s1=2, s2=4, s3=5>	scalene	scalene	pass
2.	<s1=5, s2=5, s3=6>	isosceles	isosceles	pass
3.	<s1=2, s2=2, s3=2>	equilateral	equilateral	pass
4.	<s1=1, s2=2, s3=6>	notriangle	notriangle	pass
5.	<s1=0, s2=0, s3=0>	Bad side	Bad side	pass

2. Edge Coverage Criteria or Condition Decision Coverage Criteria (ECC/CDCC):

Autotester supports the Edge Coverage (EC). Test cases generated by the autotester are not minimized.

Control Flow Graph (CFG)	Code under Test	Test Paths generated by Autotester

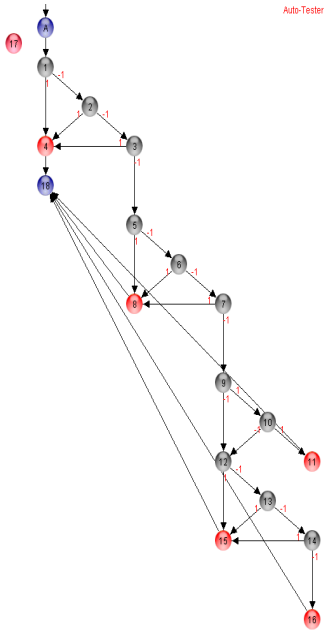
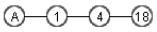
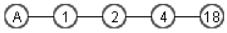
	A public static Type TriangleTest (int s1 , int s2 , int s3) {
	1 if (s1 <= 0 || 2 s2 <= 0 || 3 s3 <= 0) {
	4 return Type . BADSIDE ;
	}
	5 else {
	if (s1 + s2 <= s3 || 6 s2 + s3 <= s1 || 7 s1 + s3 <= s2) {
	8 return Type . NOTRIANGLE ;
	}
	9 else {
	if (s1 == s2 && 10 s2 == s3) {
	11 return Type . EQUILATERAL ;
	}
	12 else {
	if (s1 == s2 || 13 s2 == s3 || 14 s1 == s3) {
	15 return Type . ISOSCELES ;
	}
	16 else {
	return Type . SCALENE ;
	}
	17 }
	}
	}
	}
	18 // end
		1 A—1—4—18
		2 A—1—2—4—18
		3 A—1—2—3—5—8—18
		4 A—1—2—3—5—6—8—18
		5 A—1—2—3—5—6—7—9—12—15—18
		6 A—1—2—3—5—6—7—9—12—13—15—18
		7 A—1—2—3—5—6—7—9—12—13—14—16—18
		8 A—1—2—3—5—6—7—9—12—13—14—15—18
		9 A—1—2—3—5—6—7—9—10—12—15—18
		10 A—1—2—3—5—6—7—9—10—11—18
		11 A—1—2—3—5—6—7—8—18
		12 A—1—2—3—4—18

Fig: Table 2

Test Cases for execution of Test Paths for Edge Coverage:

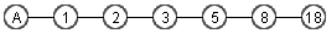
1 

① if(s1 <= 0) true
s1 = -1

2 

① if(s1 <= 0) false
s1 = 1

② if(|| s2 <= 0) true

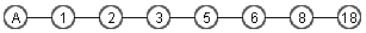
3 

① if(s1 <= 0) false
s1 = 1

② if(|| s2 <= 0) false

③ if(|| s3 <= 0) false

⑤ if(s1 + s2 <= s3) true

4 

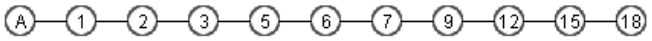
① if(s1 <= 0) false
s1 = 1

② if(|| s2 <= 0) false

③ if(|| s3 <= 0) false

⑤ if(s1 + s2 <= s3) false

⑥ if(|| s2 + s3 <= s1) true

5 

---

① if(s1 <= 0) false
s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

---

⑥ if(|| s2 + s3 <= s1) false

---

⑦ if(|| s1 + s3 <= s2) false

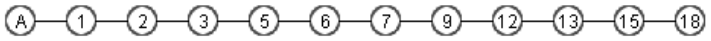
---

⑨ if(s1 == s2) false

---

⑫ if(s1 == s2) true

---

6 

---

① if(s1 <= 0) false
s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

---

⑥ if(|| s2 + s3 <= s1) false

---

⑦ if(|| s1 + s3 <= s2) false

---

⑨ if(s1 == s2) false

---

⑫ if(s1 == s2) false

---

⑬ if(|| s2 == s3) true

---

7 Ⓐ—①—②—③—⑤—⑥—⑦—⑨—⑫—⑬—⑭—⑯—⑱

---

① if(s1 <= 0) false
s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

---

⑥ if(|| s2 + s3 <= s1) false

---

⑦ if(|| s1 + s3 <= s2) false

---

⑨ if(s1 == s2) false

---

⑫ if(s1 == s2) false

---

⑬ if(|| s2 == s3) false

---

⑭ if(|| s1 == s3) false

---

8 Ⓐ—①—②—③—⑤—⑥—⑦—⑨—⑫—⑬—⑭—⑮—⑱

---

① if(s1 <= 0) false
s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

---

⑥ if(|| s2 + s3 <= s1) false

---

⑦ if(|| s1 + s3 <= s2) false

---

⑨ if(s1 == s2) false

---

⑫ if(s1 == s2) false

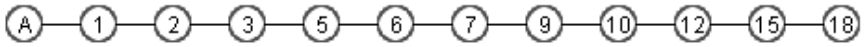
---

⑬ if(|| s2 == s3) false

---

⑭ if(|| s1 == s3) true

---

9 

① if(s1 <= 0) false
s1 = 1

② if(|| s2 <= 0) false

③ if(|| s3 <= 0) false

⑤ if(s1 + s2 <= s3) false

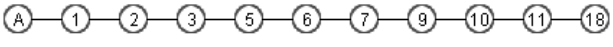
⑥ if(|| s2 + s3 <= s1) false

⑦ if(|| s1 + s3 <= s2) false

⑨ if(s1 == s2) true

⑩ if(&& s2 == s3) false

⑫ if(s1 == s2) true

10 

① if(s1 <= 0) false
s1 = 1

② if(|| s2 <= 0) false

③ if(|| s3 <= 0) false

⑤ if(s1 + s2 <= s3) false

⑥ if(|| s2 + s3 <= s1) false

⑦ if(|| s1 + s3 <= s2) false

⑨ if(s1 == s2) true

⑩ if(&& s2 == s3) true

11 A—1—2—3—5—6—7—8—18

---

1 if(s1 <= 0) false

s1 = 1

---

2 if(|| s2 <= 0) false

---

3 if(|| s3 <= 0) false

---

5 if(s1 + s2 <= s3) false

---

6 if(|| s2 + s3 <= s1) false

---

7 if(|| s1 + s3 <= s2) true

---

12 A—1—2—3—4—18

---

1 if(s1 <= 0) false

s1 = 1

---

2 if(|| s2 <= 0) false

---

3 if(|| s3 <= 0) true

Fig: Test Cases Table

Test Path	Test Case	Expected Output	Observed Output	Verdict
1.	<s1=-1, s2=2, s3=3>	badside	badside	pass
2.	<s1=2, s2=0, s3=5>	badside	badside	pass
3.	<s1=2, s2=1, s3=5>	notriangle	notriangle	pass
4.	<s1=5, s2=6, s3=5>	isosceles	isosceles	pass
5.	<s1=5, s2=5, s3=6>	isosceles	isosceles	pass
6.	<s1=6, s2=5, s3=5>	isosceles	isosceles	pass
7.	<s1=4, s2=5, s3=6>	scalene	scalene	pass
8.	<s1=4, s2=5, s3=4>	isosceles	isosceles	pass
9.	<s1=1, s2=6, s3=2>	notriangle	notriangle	pass
10.	<s1=5, s2=5, s3=5>	equilateral	equilateral	pass
11.	<s1=2, s2=5, s3=1>	Notriangle	notriangle	pass
12.	<s1=4, s2=5, s3=0>	Notriangle	notriangle	pass

3. Predicate Coverage Criteria or Decision Coverage Criteria (PCC/DCC):

Autotester supports the PC. However, test cases generated by the autotester are not minimized.

Control Flow Graph (CFG)	Code under Test	Test Paths generated by Autotester

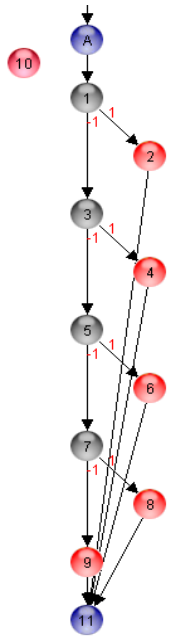
	A public static Type TriangleTest (int s1 , int s2 , int s3) {
	1 if (s1 <= 0 || s2 <= 0 || s3 <= 0) {
	2 return Type . BADSIDE ;
	}
	3 else {
	if (s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) {
	4 return Type . NOTRIANGLE ;
	}
	5 else {
	if (s1 == s2 && s2 == s3) {
	6 return Type . EQUILATERAL ;
	}
	7 else {
	if (s1 == s2 || s2 == s3 || s1 == s3) {
	8 return Type . ISOSCELES ;
	}
	9 else {
	return Type . SCALENE ;
	}
	10 }
	}
	}
	}
	11 // end
		1 A 1 3 5 7 9 11
		2 A 1 3 5 7 8 11
		3 A 1 3 5 6 11
		4 A 1 3 4 11
		5 A 1 2 11

Fig: Table 3

Test Cases for execution of Test Paths for Predicate Coverage:

1 

---

1 if (s1 <= 0 || s2 <= 0 || s3 <= 0) false

---

3 if (s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) false

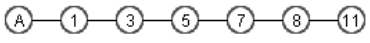
---

5 if (s1 == s2 && s2 == s3) false

---

7 if (s1 == s2 || s2 == s3 || s1 == s3) false

---

2 

---

1 if (s1 <= 0 || s2 <= 0 || s3 <= 0) false

---

3 if (s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) false

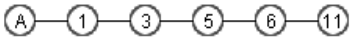
---

5 if (s1 == s2 && s2 == s3) false

---

7 if (s1 == s2 || s2 == s3 || s1 == s3) true

---

3 

---

1 if (s1 <= 0 || s2 <= 0 || s3 <= 0) false

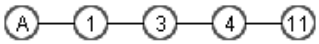
---

3 if (s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) false

---

5 if (s1 == s2 && s2 == s3) true

---

4 

---

1 if (s1 <= 0 || s2 <= 0 || s3 <= 0) false

---

3 if (s1 + s2 <= s3 || s2 + s3 <= s1 || s1 + s3 <= s2) true

Fig: Test cases table

Test Path	Test Case	Expected Output	Observed Output	Verdict
3.	<s1=1, s2=1, s3=1>	equilateral	equilateral	pass
1.	<s1=2, s2=4, s3=5>	Scalene	scalene	pass
4.	<s1=2, s2=2, s3=4>	notriangle	notriangle	pass
2.	<s1=9, s2=3, s3=9>	Isosceles	isosceles	pass
5.	<s1=-2, s2=4, s3=3>	Bad side	Bad side	pass

4. Clause Coverage Criteria or Condition Coverage Criteria (CCC/CCC):

Autotester supports the Clause Coverage (CC). However, test cases generated by the autotester are not minimized.

Control Flow Graph (CFG)	Code under Test	Test Paths generated by Autotester

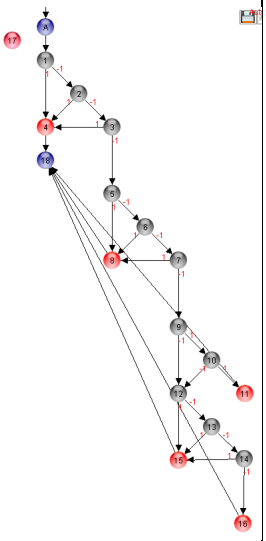
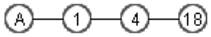
	A public static Type TriangleTest (int s1 , int s2 , int s3) {
	1 if (s1 <= 0 || 2 s2 <= 0 || 3 s3 <= 0) {
	4 return Type . BADSIDE ;
	}
	5 else {
	if (s1 + s2 <= s3 || 6 s2 + s3 <= s1 || 7 s1 + s3 <= s2) {
	8 return Type . NOTRIANGLE ;
	}
	9 else {
	if (s1 == s2 && 10 s2 == s3) {
	11 return Type . EQUILATERAL ;
	}
	12 else {
	if (s1 == s2 || 13 s2 == s3 || 14 s1 == s3) {
	15 return Type . ISOSCELES ;
	}
	16 else {
	return Type . SCALENE ;
	}
	17 }
	}
	}
	}
	18 // end
		1 A—1—4—18
		2 A—1—2—4—18
		3 A—1—2—3—5—8—18
		4 A—1—2—3—5—6—8—18
		5 A—1—2—3—5—6—7—9—12—15—18
		6 A—1—2—3—5—6—7—9—12—13—15—18
		7 A—1—2—3—5—6—7—9—12—13—14—16—18
		8 A—1—2—3—5—6—7—9—12—13—14—15—18
		9 A—1—2—3—5—6—7—9—10—12—15—18
		10 A—1—2—3—5—6—7—9—10—11—18
		11 A—1—2—3—5—6—7—8—18
		12 A—1—2—3—4—18

Fig: Table 4

Test Cases for execution of Test Paths for Clause Coverage:

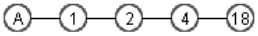
1 

---

① if (s1 <= 0) true

s1 = -1

---

2 

---

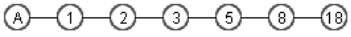
① if (s1 <= 0) false

s1 = 1

---

② if (|| s2 <= 0) true

---

3 

---

① if (s1 <= 0) false

s1 = 1

---

② if (|| s2 <= 0) false

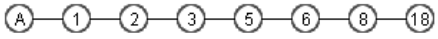
---

③ if (|| s3 <= 0) false

---

⑤ if (s1 + s2 <= s3) true

---

4 

---

① if (s1 <= 0) false

s1 = 1

---

② if (|| s2 <= 0) false

---

③ if (|| s3 <= 0) false

---

⑤ if (s1 + s2 <= s3) false

---

⑥ if (|| s2 + s3 <= s1) true

---

5 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(12)—(15)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) false

---

(12) if(s1 == s2) true

---

6 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(12)—(13)—(15)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) false

---

(12) if(s1 == s2) false

---

(13) if(|| s2 == s3) true

---

7 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(12)—(13)—(14)—(16)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) false

---

(12) if(s1 == s2) false

---

(13) if(|| s2 == s3) false

---

(14) if(|| s1 == s3) false

---

8 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(12)—(13)—(14)—(15)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) false

---

(12) if(s1 == s2) false

---

(13) if(|| s2 == s3) false

---

(14) if(|| s1 == s3) true

---

9 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(10)—(12)—(15)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) true

---

(10) if(&& s2 == s3) false

---

(12) if(s1 == s2) true

---

10 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(10)—(11)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

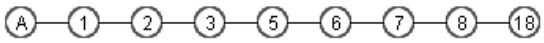
---

(9) if(s1 == s2) true

---

(10) if(&& s2 == s3) true

---

11 

---

① if(s1 <= 0) false

s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

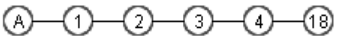
---

⑥ if(|| s2 + s3 <= s1) false

---

⑦ if(|| s1 + s3 <= s2) true

---

12 

---

① if(s1 <= 0) false

s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) true

Fig: Test Cases Table

Test Path	Test Case	Expected Output	Observed Output	Verdict
1.	<s1=-1, s2=2, s3=3>	badside	badside	pass
2.	<s1=0, s2=4, s3=5>	badside	badside	pass
3.	<s1=4, s2=2, s3=2>	notriangle	notriangle	pass
4.	<s1=5, s2=2, s3=1>	notriangle	notriangle	pass
5.	<s1=5, s2=5, s3=6>	isosceles	isosceles	pass
6.	<s1=6, s2=5, s3=5>	isosceles	isosceles	pass
7.	<s1=4, s2=5, s3=6>	scalene	scalene	pass
8.	<s1=4, s2=5, s3=4>	isosceles	isosceles	pass
9.	<s1=2, s2=2, s3=12>	isosceles	isosceles	pass
10.	<s1=5, s2=5, s3=5>	equilateral	equilateral	pass
11.	<s1=2, s2=5, s3=1>	notriangle	notriangle	pass
12.	<s1=4, s2=5, s3=0>	badside	badside	pass

.

5. Restricted Active Clause Coverage or Multi Condition Decision Coverage (RACC/MCDC):

Autotester supports the RACC. It doesn't generate the test cases however it generates the test paths for the RACC.

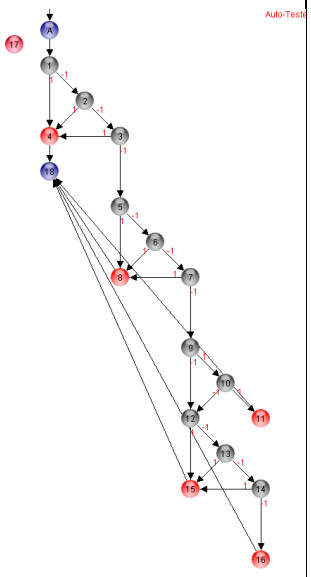
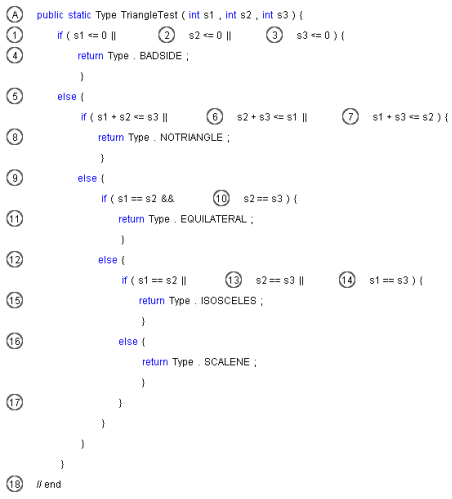
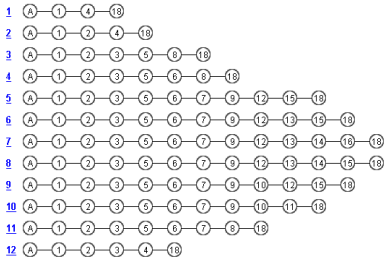
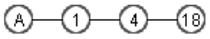
Control Flow Graph (CFG)	Code under Test	Test Paths generated by Autotester
		

Fig: Table 5

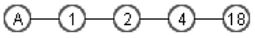
Test Cases for execution of Test Paths for Restricted Active Clause Coverage:

1 

---

1 if(s1 <= 0) true
s1 = -1

---

2 

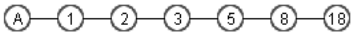
---

1 if(s1 <= 0) false
s1 = 1

---

2 if(|| s2 <= 0) true

---

3 

---

1 if(s1 <= 0) false
s1 = 1

---

2 if(|| s2 <= 0) false

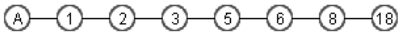
---

3 if(|| s3 <= 0) false

---

5 if(s1 + s2 <= s3) true

---

4 

---

1 if(s1 <= 0) false
s1 = 1

---

2 if(|| s2 <= 0) false

---

3 if(|| s3 <= 0) false

---

5 if(s1 + s2 <= s3) false

---

6 if(|| s2 + s3 <= s1) true

---

5 

---

① if(s1 <= 0) false
s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

---

⑥ if(|| s2 + s3 <= s1) false

---

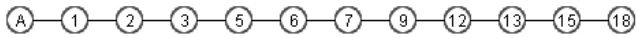
⑦ if(|| s1 + s3 <= s2) false

---

⑨ if(s1 == s2) false

---

⑫ if(s1 == s2) true

6 

---

① if(s1 <= 0) false
s1 = 1

---

② if(|| s2 <= 0) false

---

③ if(|| s3 <= 0) false

---

⑤ if(s1 + s2 <= s3) false

---

⑥ if(|| s2 + s3 <= s1) false

---

⑦ if(|| s1 + s3 <= s2) false

---

⑨ if(s1 == s2) false

---

⑫ if(s1 == s2) false

---

⑬ if(|| s2 == s3) true

---

7 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(12)—(13)—(14)—(16)—(18)

(1) if(s1 <= 0) false
s1 = 1

(2) if(|| s2 <= 0) false

(3) if(|| s3 <= 0) false

(5) if(s1 + s2 <= s3) false

(6) if(|| s2 + s3 <= s1) false

(7) if(|| s1 + s3 <= s2) false

(9) if(s1 == s2) false

(12) if(s1 == s2) false

(13) if(|| s2 == s3) false

(14) if(|| s1 == s3) false

8 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(12)—(13)—(14)—(15)—(18)

(1) if(s1 <= 0) false
s1 = 1

(2) if(|| s2 <= 0) false

(3) if(|| s3 <= 0) false

(5) if(s1 + s2 <= s3) false

(6) if(|| s2 + s3 <= s1) false

(7) if(|| s1 + s3 <= s2) false

(9) if(s1 == s2) false

(12) if(s1 == s2) false

(13) if(|| s2 == s3) false

(14) if(|| s1 == s3) true

9 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(10)—(12)—(15)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) true

---

(10) if(&& s2 == s3) false

---

(12) if(s1 == s2) true

---

10 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(9)—(10)—(11)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) false

---

(9) if(s1 == s2) true

---

(10) if(&& s2 == s3) true

---

11 (A)—(1)—(2)—(3)—(5)—(6)—(7)—(8)—(18)

---

(1) if(s1 <= 0) false
s1 = 1

---

(2) if(|| s2 <= 0) false

---

(3) if(|| s3 <= 0) false

---

(5) if(s1 + s2 <= s3) false

---

(6) if(|| s2 + s3 <= s1) false

---

(7) if(|| s1 + s3 <= s2) true

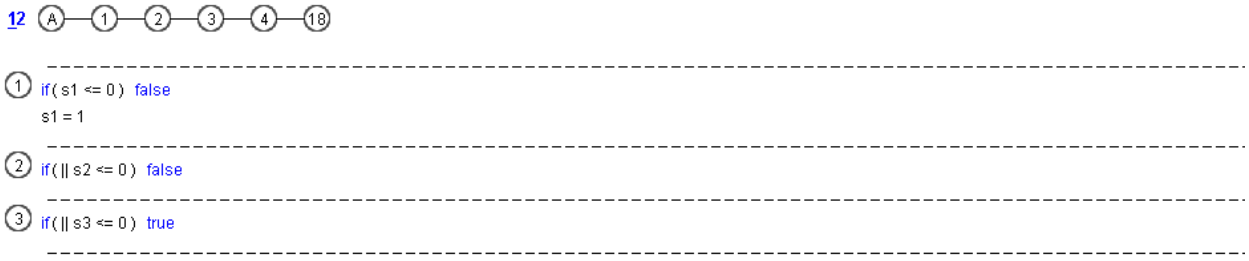


Fig: Test Cases Table

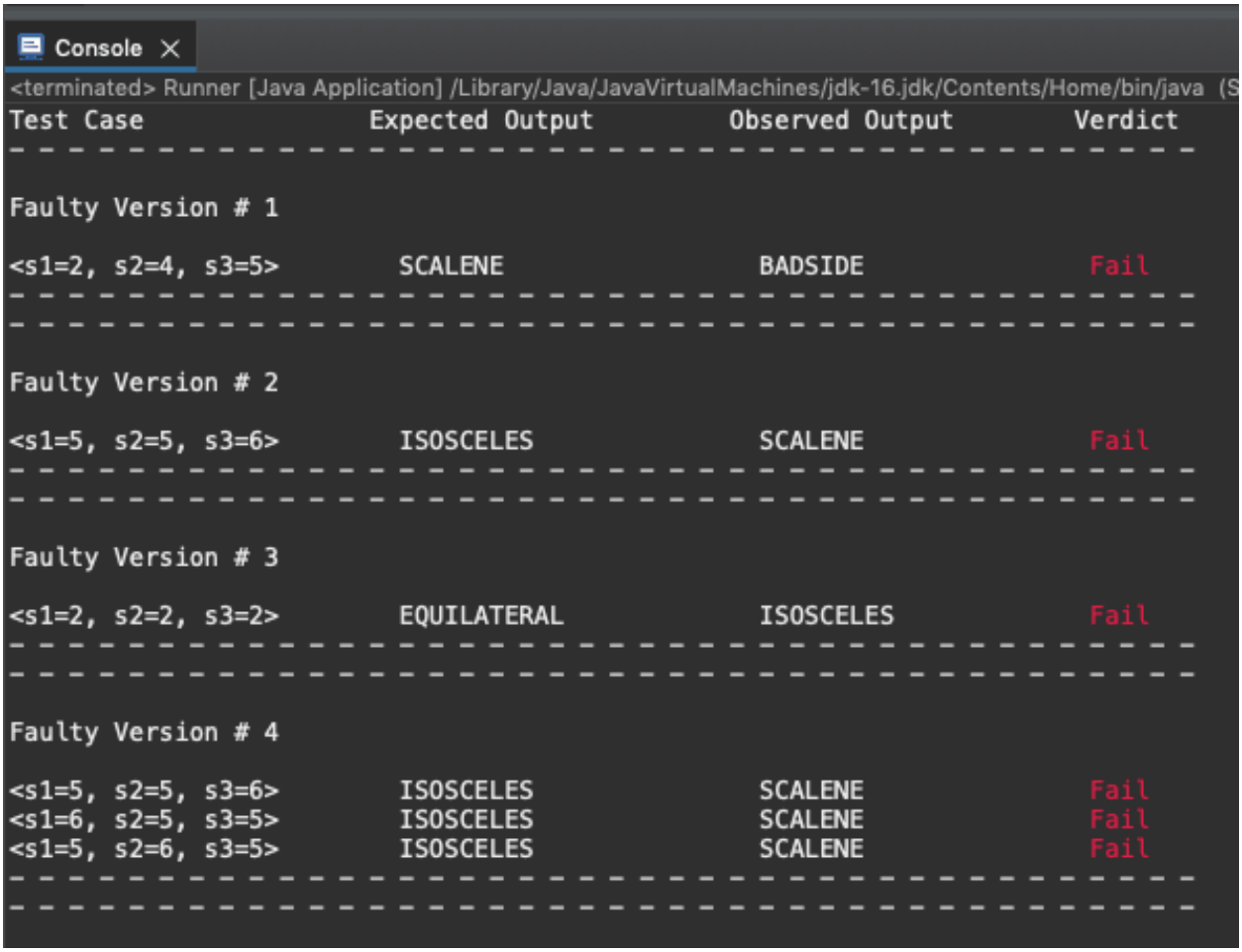
Test Path	Test Case	Expected Output	Observed Output	Verdict
1.	<s1=-1, s2=2, s3=3>	badside	badside	pass
2.	<s1=2, s2=0, s3=5>	badside	badside	pass
3.	<s1=2, s2=4, s3=2>	notriangle	notriangle	pass
4.	<s1=5, s2=2, s3=1>	notriangle	notriangle	pass
5.	<s1=5, s2=5, s3=6>	Bad side	Bad side	pass
6.	<s1=6, s2=5, s3=5>	isosceles	isosceles	pass
7.	<s1=4, s2=5, s3=6>	scalene	scalene	pass
8.	<s1=4, s2=5, s3=4>	isosceles	isosceles	pass
9.	<s1=2, s2=2, s3=12>	notriangle	notriangle	pass
10.	<s1=5, s2=5, s3=5>	equilateral	equilateral	pass
11.	<s1=2, s2=5, s3=1>	Notriangle	notriangle	pass
12.	<s1=4, s2=5, s3=0>	Notriangle	notriangle	pass

Execution of Test Cases on 14 Fault versions Jar Files:

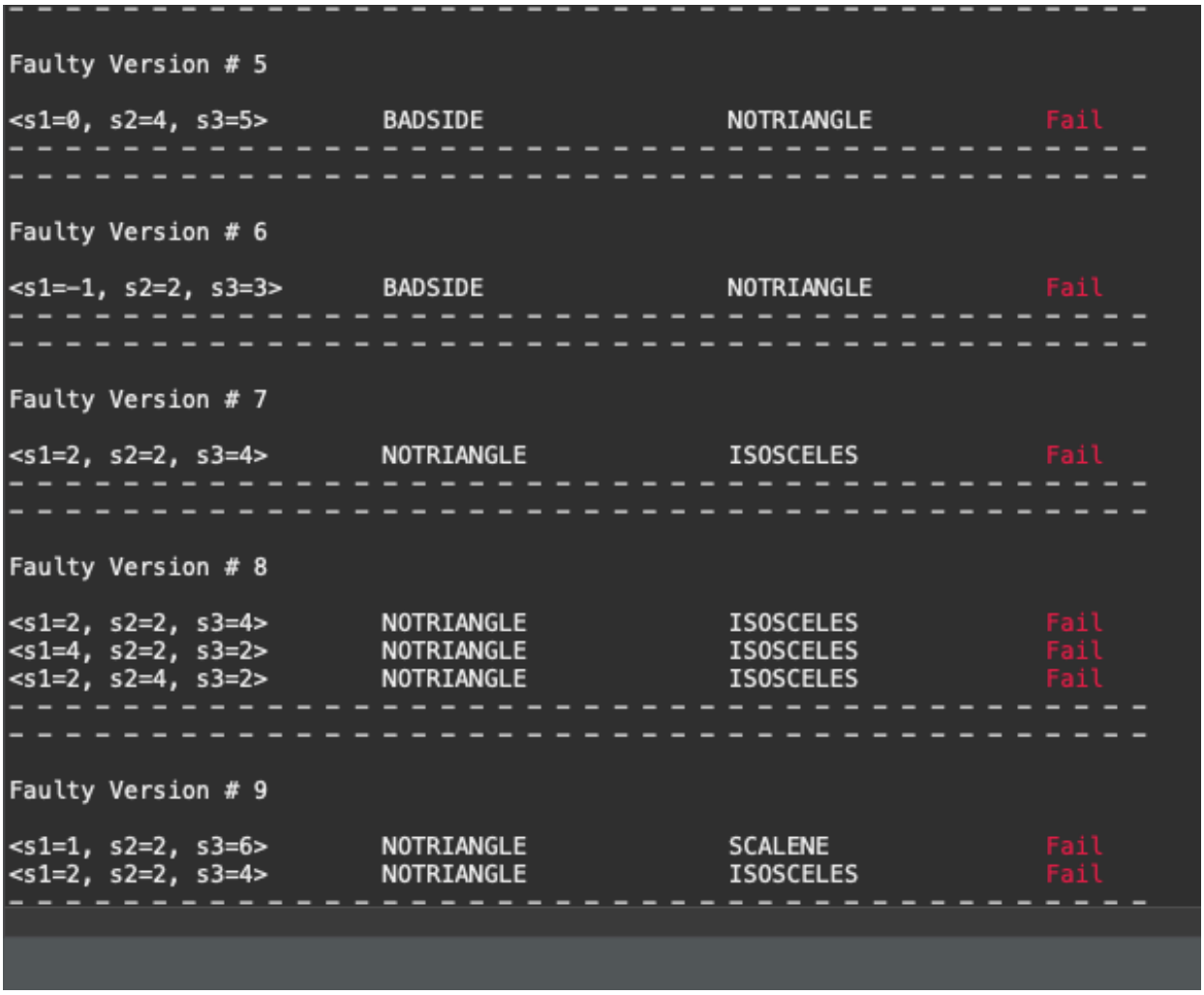
I executed all the test cases generated by the autotester on the 14 faulty versions of original program one by one. Following are the result of the execution.

Results Table:

1-4 Findings:



5-9 Findings:


Faulty Version # 5
<s1=0, s2=4, s3=5> BADSIDE NOTRIANGLE Fail
Faulty Version # 6
<s1=-1, s2=2, s3=3> BADSIDE NOTRIANGLE Fail
Faulty Version # 7
<s1=2, s2=2, s3=4> NOTRIANGLE ISOSCELES Fail
Faulty Version # 8
<s1=2, s2=2, s3=4> NOTRIANGLE ISOSCELES Fail
<s1=4, s2=2, s3=2> NOTRIANGLE ISOSCELES Fail
<s1=2, s2=4, s3=2> NOTRIANGLE ISOSCELES Fail
Faulty Version # 9
<s1=1, s2=2, s3=6> NOTRIANGLE SCALENE Fail
<s1=2, s2=2, s3=4> NOTRIANGLE ISOSCELES Fail

10-12 Findings: