Assignment on jCUTE

Submitted by: -

MTech, Software Engineering

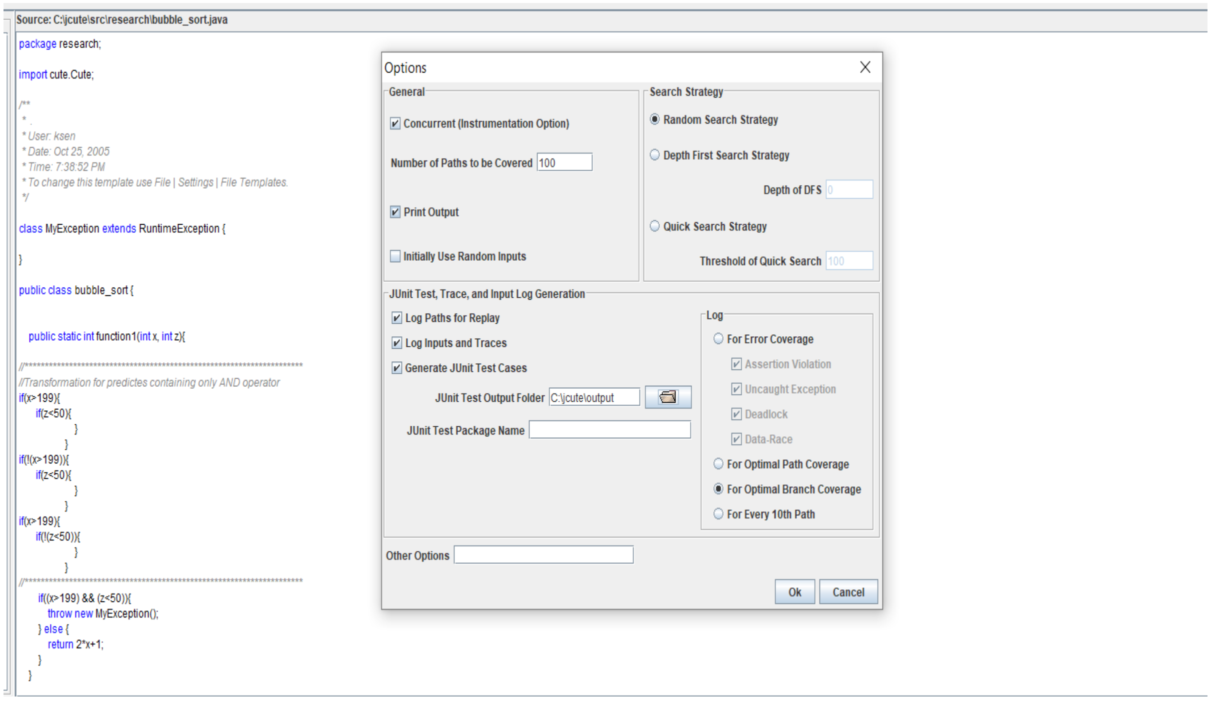
Sub – Software Testing Lab

National Institute of Technology

Rourkela

1) bubble_sort.java

first, we try with following settings -



Source: C:\jcute\src\research\bubble_sort.java

```
package research;

import cute.Cute;

/**
 * .
 * User: ksen
 * Date: Oct 25, 2005
 * Time: 7:38:52 PM
 * To change this template use File | Settings | File Templates.
 */

class MyException extends RuntimeException {

}

public class bubble_sort {


   public static int function1(int x, int z){

//*************************************************************************
//Transformation for predictes containing only AND operator
if(x>199){
     if(z<50){
               }
            }
if(!(x>199)){
     if(z<50){
               }
            }
if(x>199){
     if(!(z<50)){
               }
            }
//*************************************************************************
      if((x>199) && (z<50)){
         throw new MyException();
      } else {
         return 2*x+1;
      }
   }
```

Options

General

- [✓] Concurrent (Instrumentation Option)
- Number of Paths to be Covered: 100
- [✓] Print Output
- [ ] Initially Use Random Inputs

Search Strategy

- [✓ Random Search Strategy]
- Depth First Search Strategy
  - Depth of DFS: 0
- Quick Search Strategy
  - Threshold of Quick Search: 100

JUnit Test, Trace, and Input Log Generation

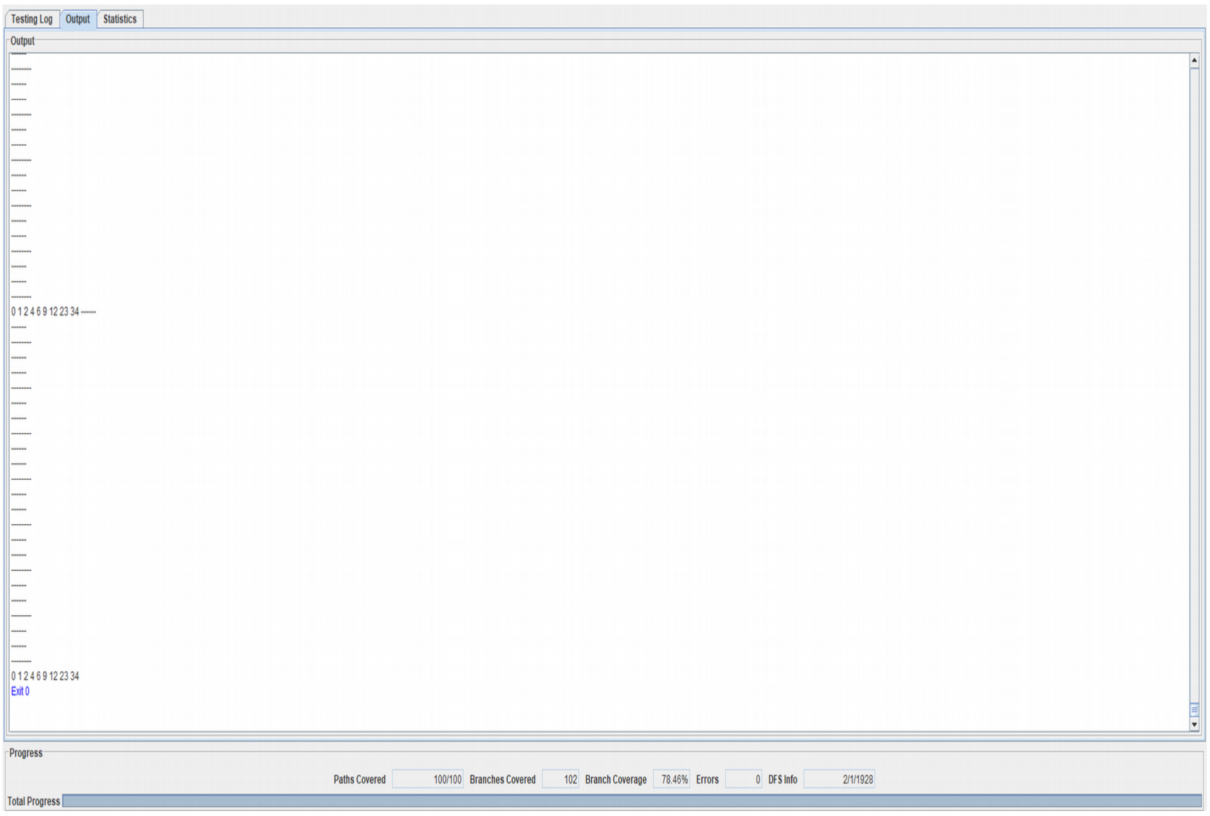
- [✓] Log Paths for Replay
- [✓] Log Inputs and Traces
- [✓] Generate JUnit Test Cases
- JUnit Test Output Folder: C:\jcute\output
- JUnit Test Package Name:

Log

- For Error Coverage
  - [✓] Assertion Violation
  - [✓] Uncaught Exception
  - [✓] Deadlock
  - [✓] Data-Race
- For Optimal Path Coverage
- [✓ For Optimal Branch Coverage]
- For Every 10th Path

Other Options:

Output :-



Output

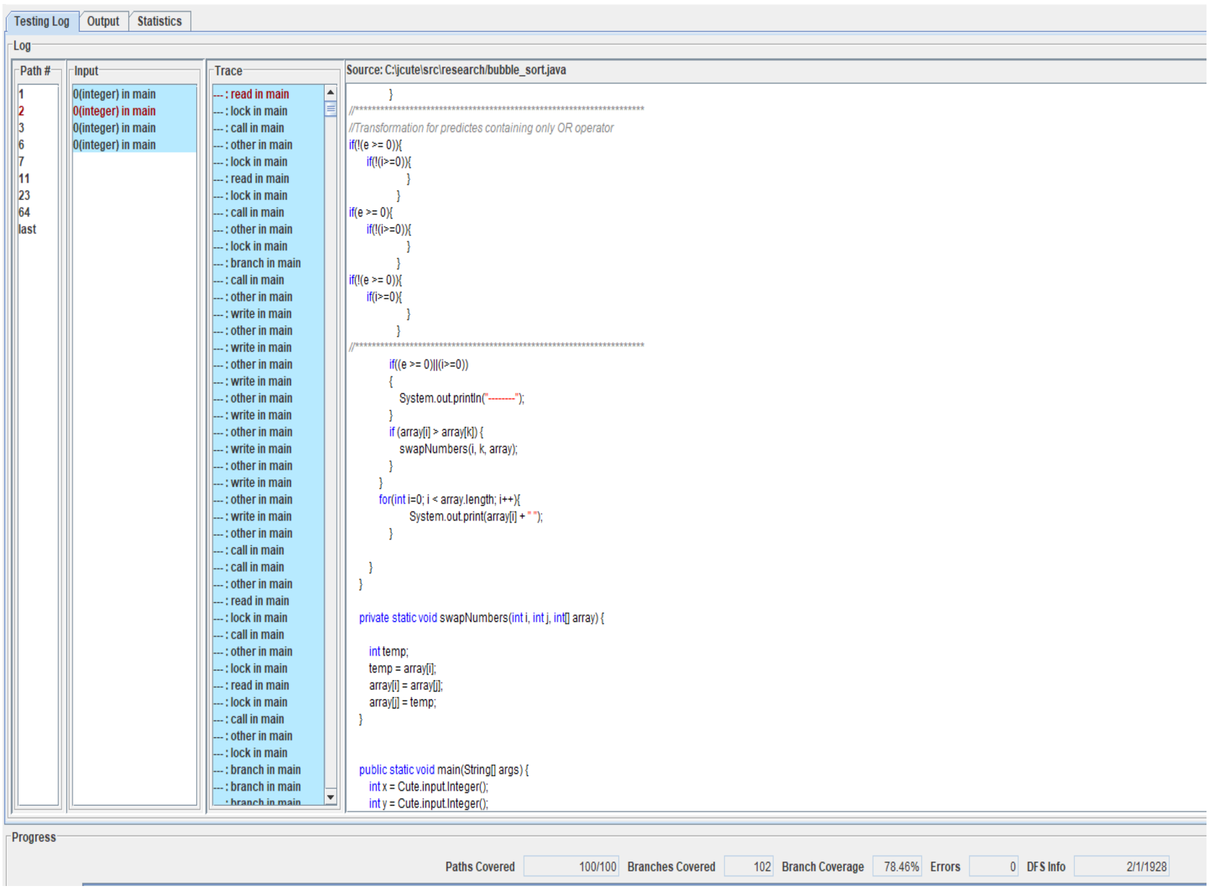
```
------
---------
------
------
---------
------
------
---------
------
------
---------
------
------
---------
------
------
---------
0 1 2 4 6 9 12 23 34 ------
------
---------
------
------
---------
------
------
---------
------
------
---------
------
------
---------
------
------
---------
------
------
---------
0 1 2 4 6 9 12 23 34
Exit 0
```

Progress

Paths Covered 100/100 Branches Covered 102 Branch Coverage 78.46% Errors 0 DFS Info 2/1/1928

Total Progress

Testing log :-



Testing Log | Output | Statistics

Log

Path #	Input
1	0(integer) in main
2	0(integer) in main
3	0(integer) in main
6	0(integer) in main
7	
11	
23	
64	
last	

Trace

```
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : branch in main
--- : call in main
--- : other in main
--- : write in main
--- : other in main
--- : write in main
--- : other in main
--- : write in main
--- : other in main
--- : write in main
--- : other in main
--- : write in main
--- : other in main
--- : write in main
--- : other in main
--- : write in main
--- : other in main
--- : call in main
--- : call in main
--- : other in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : branch in main
--- : branch in main
--- : branch in main
```

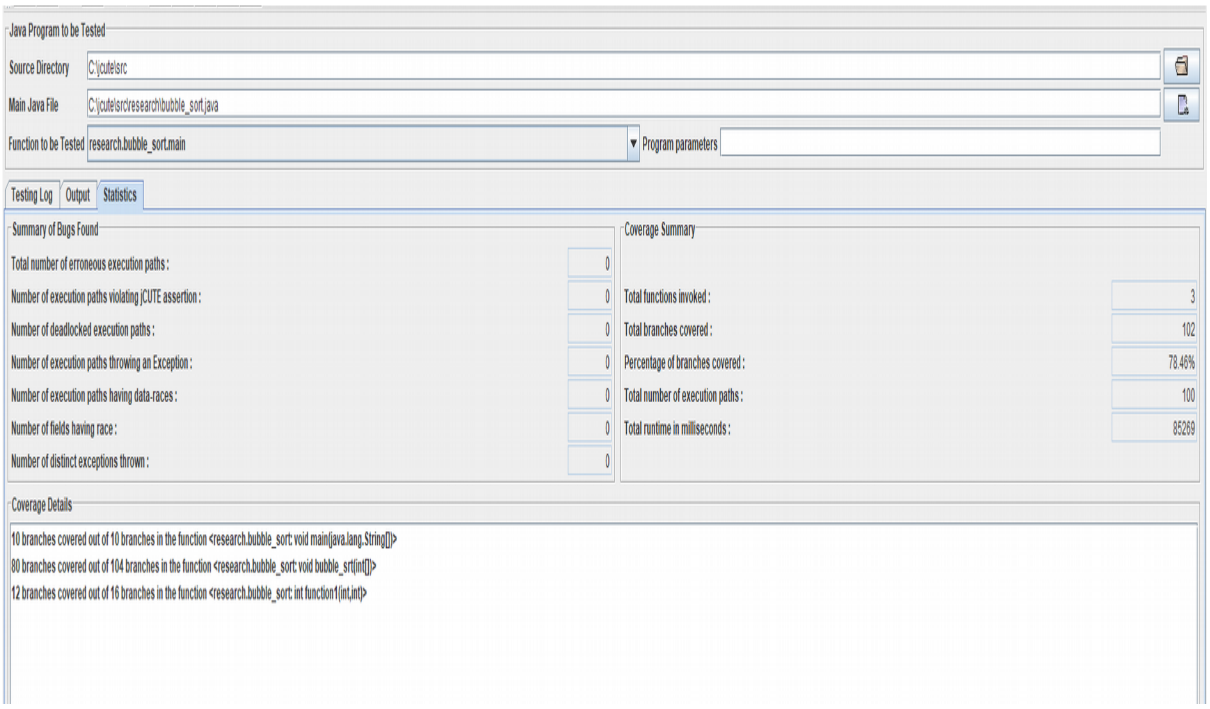
Source: C:\jcute\src\research/bubble_sort.java

```
                }
//*******************************************************************
//Transformation for predictes containing only OR operator
if(!(e >= 0)){
     if(!(i>=0)){
                }
             }
if(e >= 0){
     if(!(i>=0)){
                }
             }
if(!(e >= 0)){
     if(i>=0){
                }
             }
//*******************************************************************
              if((e >= 0)||(i>=0))
              {
                 System.out.println("--------");
              }
              if (array[i] > array[k]) {
                 swapNumbers(i, k, array);
              }
            }
            for(int i=0; i < array.length; i++){
                   System.out.print(array[i] + " ");
              }


        }
    }

    private static void swapNumbers(int i, int j, int[] array) {

        int temp;
        temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }


    public static void main(String[] args) {
        int x = Cute.input.Integer();
        int y = Cute.input.Integer();
```

Progress

Paths Covered 100/100 Branches Covered 102 Branch Coverage 78.46% Errors 0 DFS Info 2/1/1928

Statistics :-



Java Program to be Tested

Source Directory	C:\jcute\src
Main Java File	C:\jcute\src\research\bubble_sort.java
Function to be Tested	research.bubble_sort.main
Program parameters	

Testing Log | Output | Statistics

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	3
Total branches covered :	102
Percentage of branches covered :	78.46%
Total number of execution paths :	100
Total runtime in milliseconds :	85269

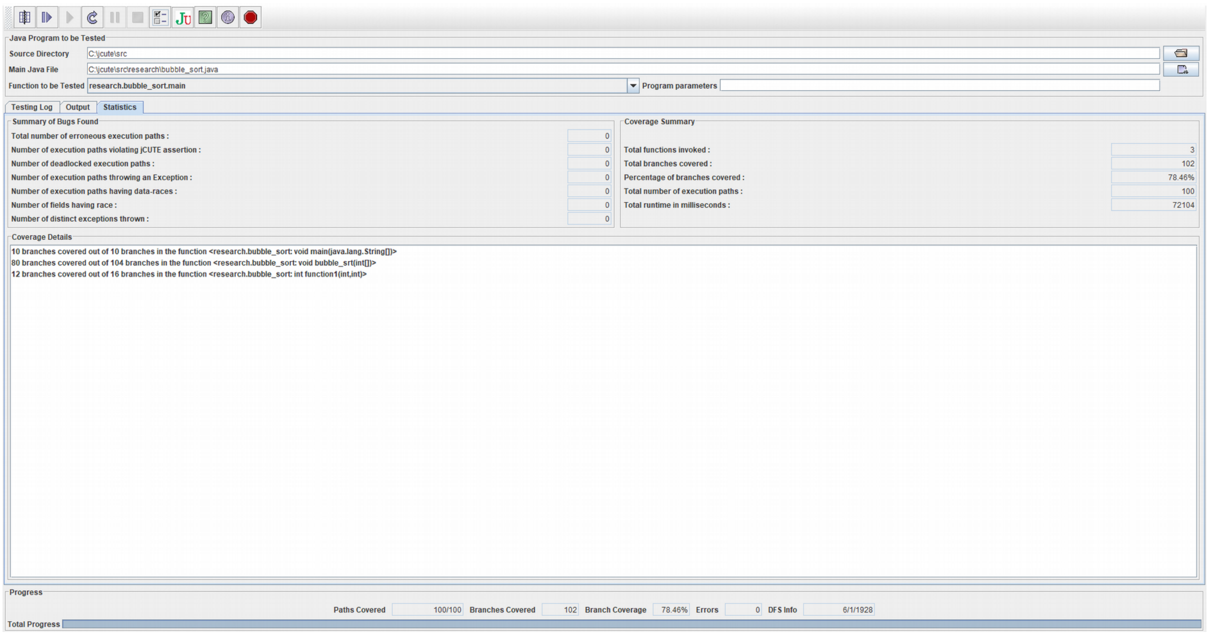
Coverage Details

```
10 branches covered out of 10 branches in the function <research.bubble_sort: void main(java.lang.String[])>
80 branches covered out of 104 branches in the function <research.bubble_sort: void bubble_srt(int[])>
12 branches covered out of 16 branches in the function <research.bubble_sort: int function1(int,int)>
```

Test case file generated –

```java
/* JUnit test case generated automatically by CUTE */
import junit.framework.*;

public class research_bubble_sort_main_Test extends TestCase
implements cute.Input {
    private Object[] input;
    private int i;

    public research_bubble_sort_main_Test(String name){
        super(name);
    }

    public boolean Boolean() {
        return ((Boolean)input[i++]).booleanValue();
    }

    public short Short() {
        return ((Short)input[i++]).shortValue();
    }

    public int Integer() {
        return ((Integer)input[i++]).intValue();
    }

    public long Long() {
        return ((Long)input[i++]).longValue();
    }

    public float Float() {
        return ((Float)input[i++]).floatValue();
    }

    public double Double() {
        return ((Double)input[i++]).doubleValue();
    }

    public char Character() {
        return ((Character)input[i++]).charValue();
    }

    public byte Byte() {
        return ((Byte)input[i++]).byteValue();
    }

    public Object Object(String type) {
        return input[i++];
    }
```

Now we use different setting-



Java Program to be Tested
Source Directory: C:\jcute\src
Main Java File: C:\jcute\src\research\bubble_sort.java
Function to be Tested: research.bubble_sort.main
Program parameters:

Testing Log | Output | Statistics

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	3
Total branches covered :	102
Percentage of branches covered :	78.46%
Total number of execution paths :	100
Total runtime in milliseconds :	72104

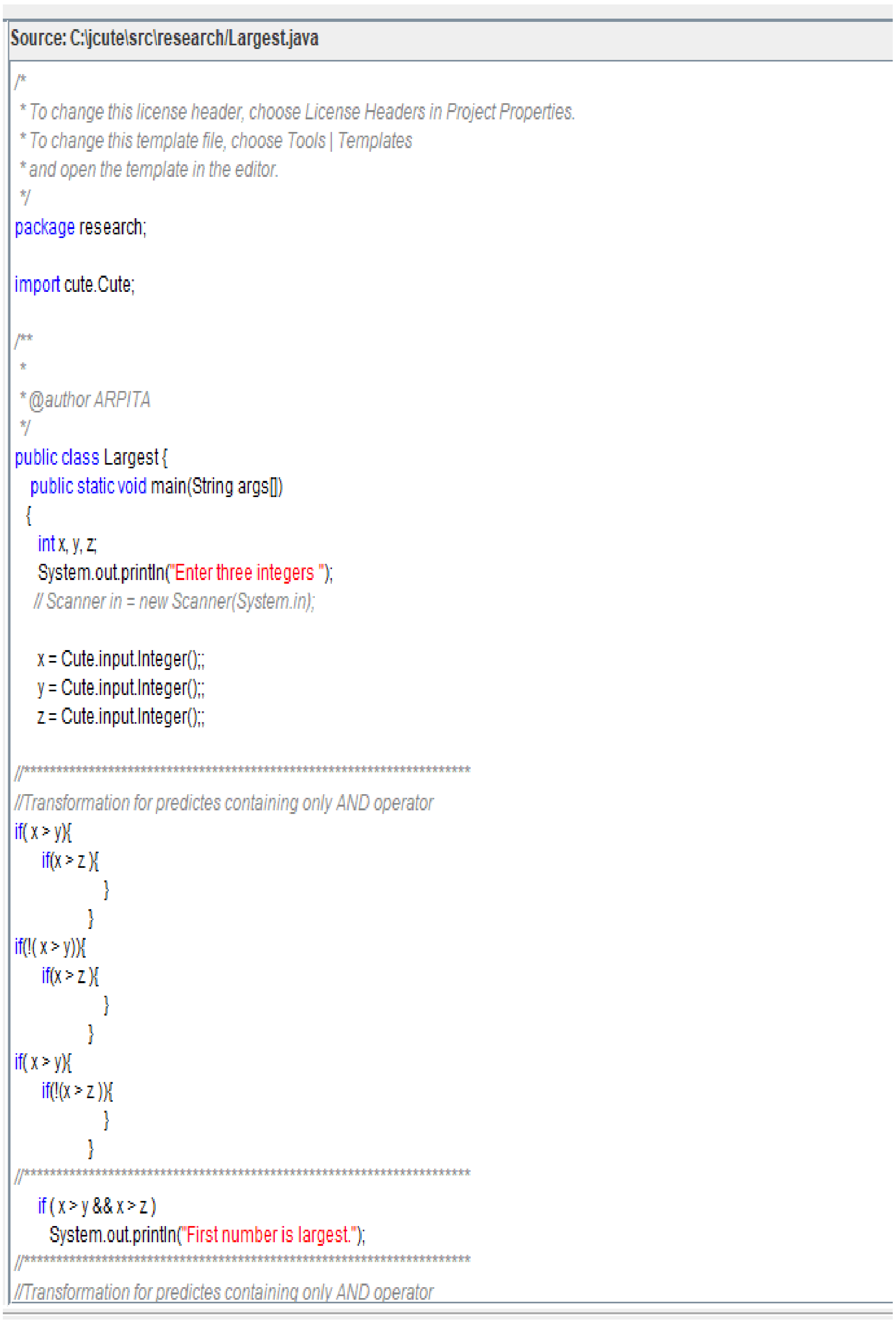
Coverage Details

```
10 branches covered out of 10 branches in the function <research.bubble_sort: void main(java.lang.String[])>
80 branches covered out of 104 branches in the function <research.bubble_sort: void bubble_srt(int[])>
12 branches covered out of 16 branches in the function <research.bubble_sort: int function1(int,int)>
```

Progress
Paths Covered 100/100 Branches Covered 102 Branch Coverage 78.46% Errors 0 DFS Info 6/1/1928
Total Progress

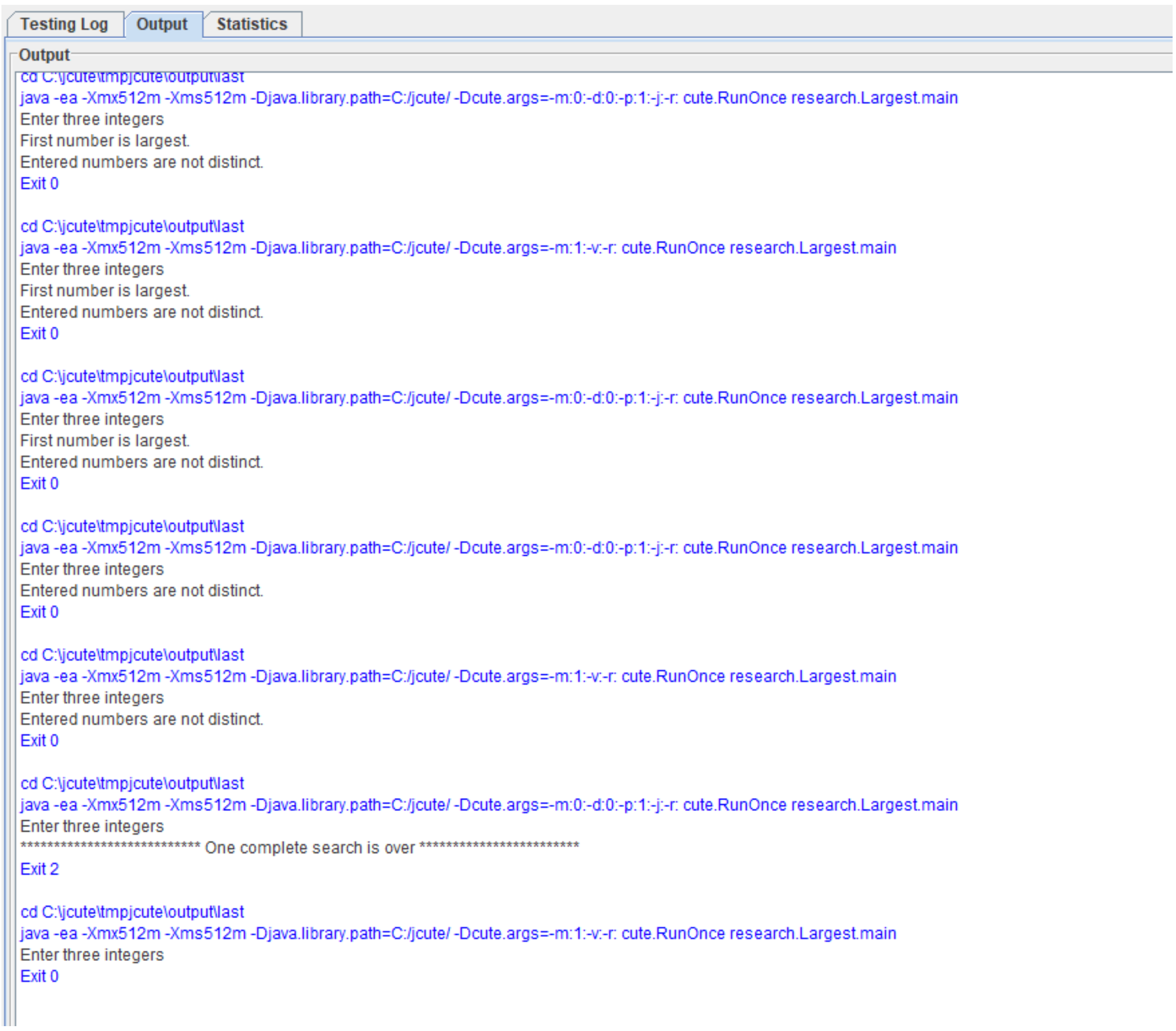
Similarly, we perform testing on different programs.

2) largest.java



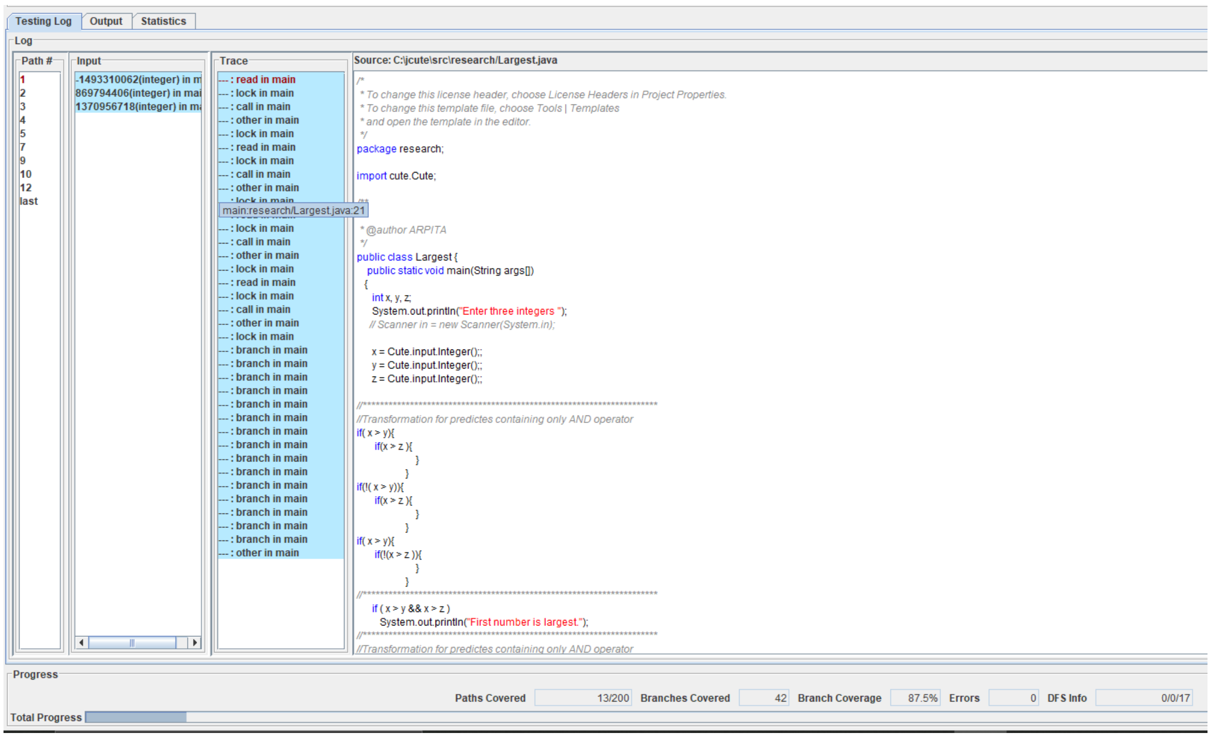
Source: C:\jcute\src\research/Largest.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package research;

import cute.Cute;

/**
 *
 * @author ARPITA
 */
public class Largest {
   public static void main(String args[])
  {
    int x, y, z;
    System.out.println("Enter three integers ");
   // Scanner in = new Scanner(System.in);

    x = Cute.input.Integer();;
    y = Cute.input.Integer();;
    z = Cute.input.Integer();;

//**********************************************************************
//Transformation for predictes containing only AND operator
if( x > y){
     if(x > z ){
                }
             }
if(!( x > y)){
     if(x > z ){
                }
             }
if( x > y){
     if(!(x > z )){
                }
             }
//**********************************************************************
    if ( x > y && x > z )
      System.out.println("First number is largest.");
//**********************************************************************
//Transformation for predictes containing only AND operator
```


Testing Log | Output | Statistics

Output

```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Largest.main
Enter three integers
First number is largest.
Entered numbers are not distinct.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.Largest.main
Enter three integers
First number is largest.
Entered numbers are not distinct.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Largest.main
Enter three integers
First number is largest.
Entered numbers are not distinct.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Largest.main
Enter three integers
Entered numbers are not distinct.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.Largest.main
Enter three integers
Entered numbers are not distinct.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Largest.main
Enter three integers
*************************** One complete search is over ************************
Exit 2

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.Largest.main
Enter three integers
Exit 0
```


Testing Log | Output | Statistics

Log

Path #
```
1
2
3
4
5
7
9
10
12
last
```

Input
```
-1493310062(integer) in m
869794406(integer) in mai
1370956718(integer) in ma
```

Trace
```
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
main:research/Largest.java:21
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : other in main
```

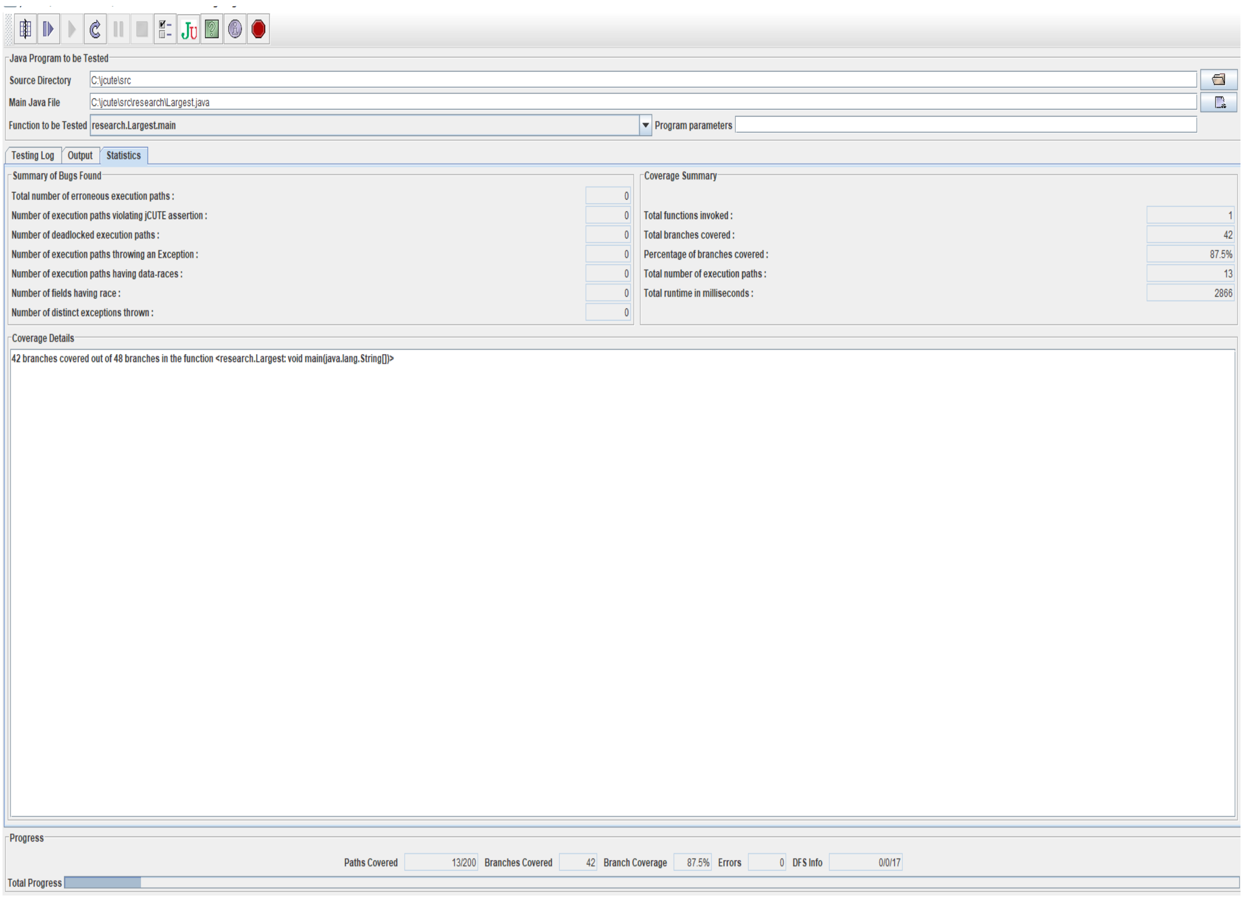
Source: C:\jcute\src\research/Largest.java

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package research;

import cute.Cute;

/**

 * @author ARPITA
 */
public class Largest {
   public static void main(String args[])
  {
    int x, y, z;
    System.out.println("Enter three integers ");
   // Scanner in = new Scanner(System.in);

    x = Cute.input.Integer();;
    y = Cute.input.Integer();;
    z = Cute.input.Integer();;

//*******************************************************************
//Transformation for predictes containing only AND operator
if( x > y){
      if(x > z ){
                }
              }
if(!( x > y)){
      if(x > z ){
                }
              }
if( x > y){
      if(!(x > z )){
                }
              }
//*******************************************************************
    if ( x > y && x > z )
      System.out.println("First number is largest.");
//*******************************************************************
//Transformation for predictes containing only AND operator
```

Progress

Paths Covered 13/200 Branches Covered 42 Branch Coverage 87.5% Errors 0 DFS Info 0/0/17

Total Progress



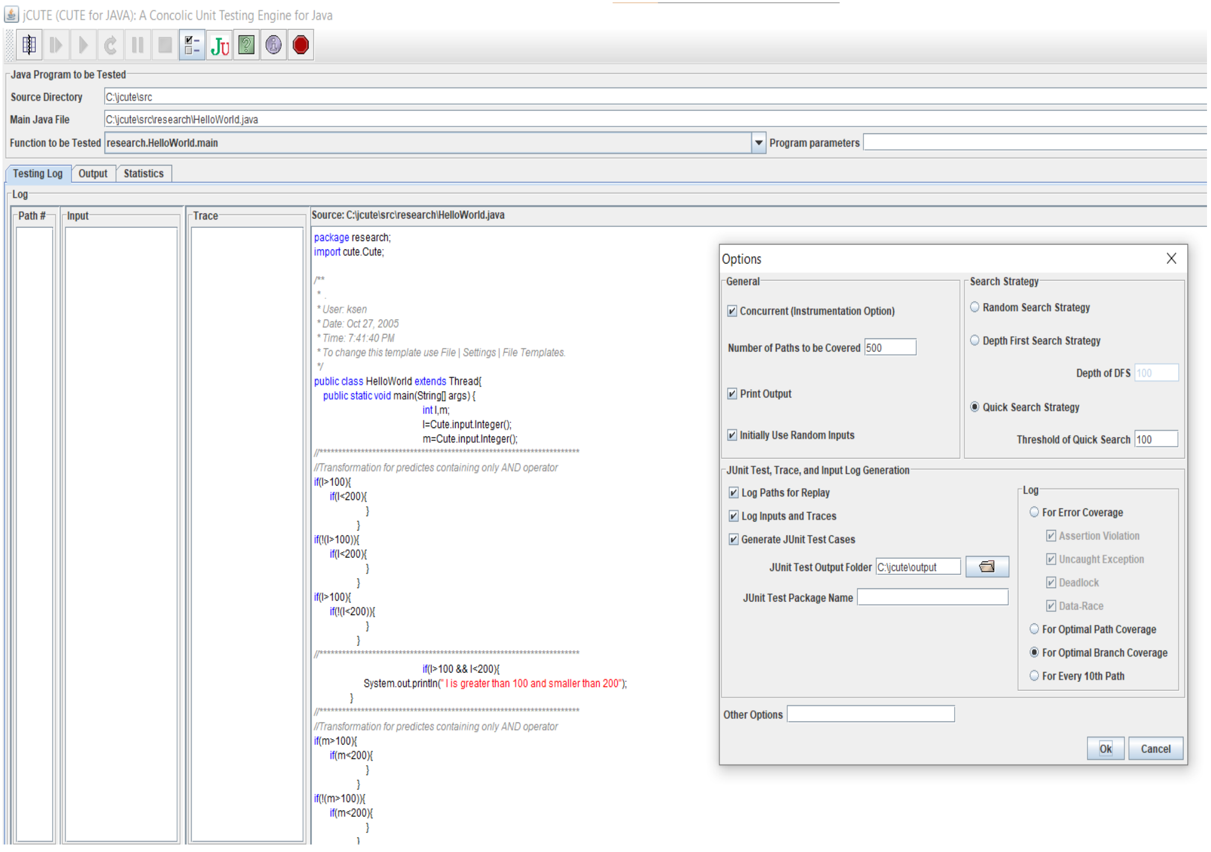
```
Java Program to be Tested
Source Directory          C:\jcute\src
Main Java File            C:\jcute\src\research\Largest.java
Function to be Tested     research.Largest.main        Program parameters

Testing Log | Output | Statistics

Summary of Bugs Found
Total number of erroneous execution paths :                    0
Number of execution paths violating jCUTE assertion :          0
Number of deadlocked execution paths :                         0
Number of execution paths throwing an Exception :              0
Number of execution paths having data-races :                  0
Number of fields having race :                                 0
Number of distinct exceptions thrown :                         0

Coverage Summary
Total functions invoked :                 1
Total branches covered :                 42
Percentage of branches covered :      87.5%
Total number of execution paths :        13
Total runtime in milliseconds :        2866

Coverage Details
42 branches covered out of 48 branches in the function <research.Largest: void main(java.lang.String[])>

Progress
Paths Covered  13/200  Branches Covered  42  Branch Coverage  87.5%  Errors  0  DFS Info  0/0/17
Total Progress
```

Test case file :-

```
/* JUnit test case generated automatically by CUTE */
import junit.framework.*;

public class research_Largest_main_Test extends TestCase
implements cute.Input {
    private Object[] input;
    private int i;

    public research_Largest_main_Test(String name){
        super(name);
    }

    public boolean Boolean() {
        return ((Boolean)input[i++]).booleanValue();
    }

    public short Short() {
        return ((Short)input[i++]).shortValue();
    }

    public int Integer() {
        return ((Integer)input[i++]).intValue();
    }

    public long Long() {
        return ((Long)input[i++]).longValue();
    }

    public float Float() {
        return ((Float)input[i++]).floatValue();
    }

    public double Double() {
        return ((Double)input[i++]).doubleValue();
    }

    public char Character() {
        return ((Character)input[i++]).charValue();
    }

    public byte Byte() {
        return ((Byte)input[i++]).byteValue();
    }

    public Object Object(String type) {
        return input[i++];
    }
```

3) HelloWorld.java

‘Quick search strategy’ was used in the setting.



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested
Source Directory: C:\jcute\src
Main Java File: C:\jcute\src\research\HelloWorld.java
Function to be Tested: research.HelloWorld.main
Program parameters:

Testing Log | Output | Statistics

Log — Path # | Input | Trace

Source: C:\jcute\src\research\HelloWorld.java

```java
package research;
import cute.Cute;

/**
 * .
 * User: ksen
 * Date: Oct 27, 2005
 * Time: 7:41:40 PM
 * To change this template use File | Settings | File Templates.
 */
public class HelloWorld extends Thread{
   public static void main(String[] args) {
                              int l,m;
                              l=Cute.input.Integer();
                              m=Cute.input.Integer();
//*******************************************************************
//Transformation for predictes containing only AND operator
if(l>100){
    if(l<200){
              }
           }
if(!(l>100)){
    if(l<200){
              }
           }
if(l>100){
    if(!(l<200)){
              }
           }
//*******************************************************************
                              if(l>100 && l<200){
              System.out.println(" l is greater than 100 and smaller than 200");
          }
//*******************************************************************
//Transformation for predictes containing only AND operator
if(m>100){
    if(m<200){
              }
           }
if(!(m>100)){
    if(m<200){
              }
           }
```

Options

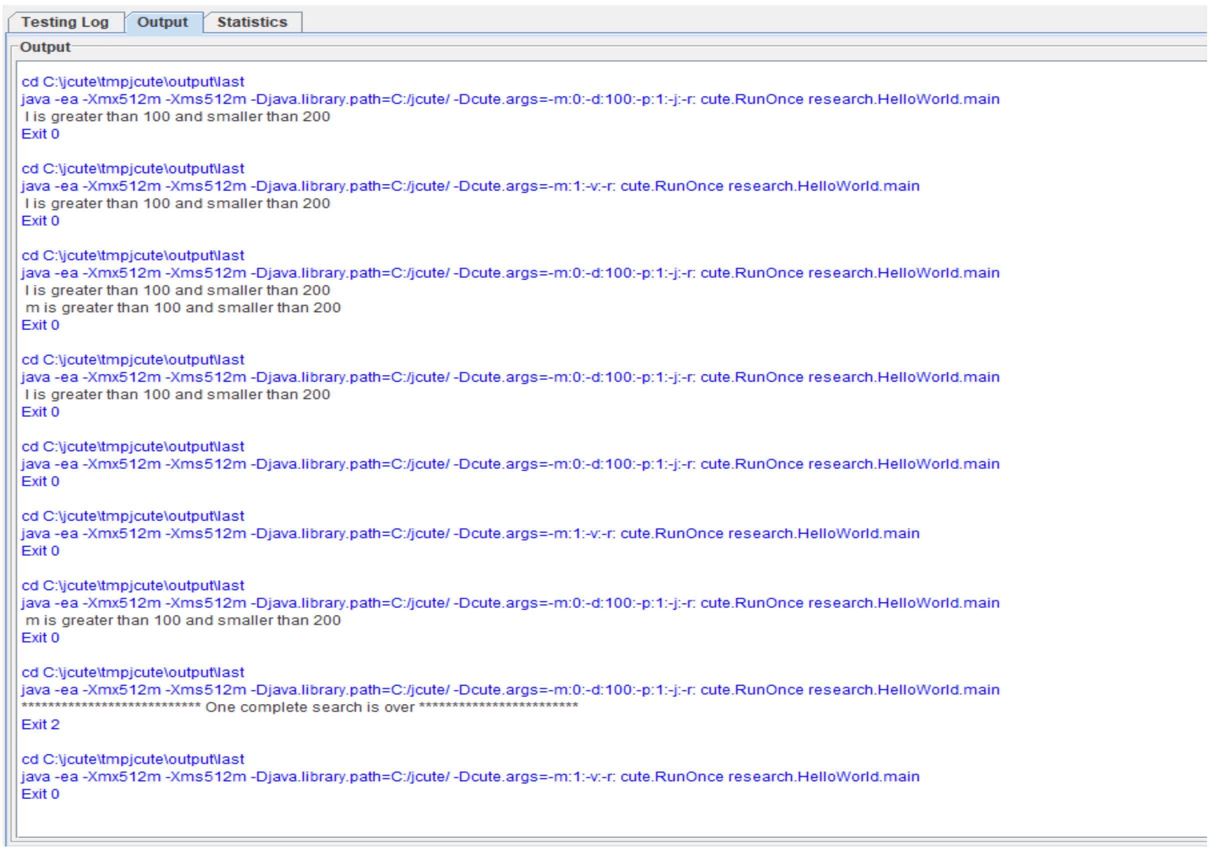
General
- [✓] Concurrent (Instrumentation Option)
- Number of Paths to be Covered: 500
- [✓] Print Output
- [✓] Initially Use Random Inputs

Search Strategy
- ○ Random Search Strategy
- ○ Depth First Search Strategy — Depth of DFS: 100
- [● Quick Search Strategy] — Threshold of Quick Search: 100

JUnit Test, Trace, and Input Log Generation
- [✓] Log Paths for Replay
- [✓] Log Inputs and Traces
- [✓] Generate JUnit Test Cases
- JUnit Test Output Folder: C:\jcute\output
- JUnit Test Package Name:

Log
- ○ For Error Coverage
  - [✓] Assertion Violation
  - [✓] Uncaught Exception
  - [✓] Deadlock
  - [✓] Data-Race
- ○ For Optimal Path Coverage
- [● For Optimal Branch Coverage]
- ○ For Every 10th Path

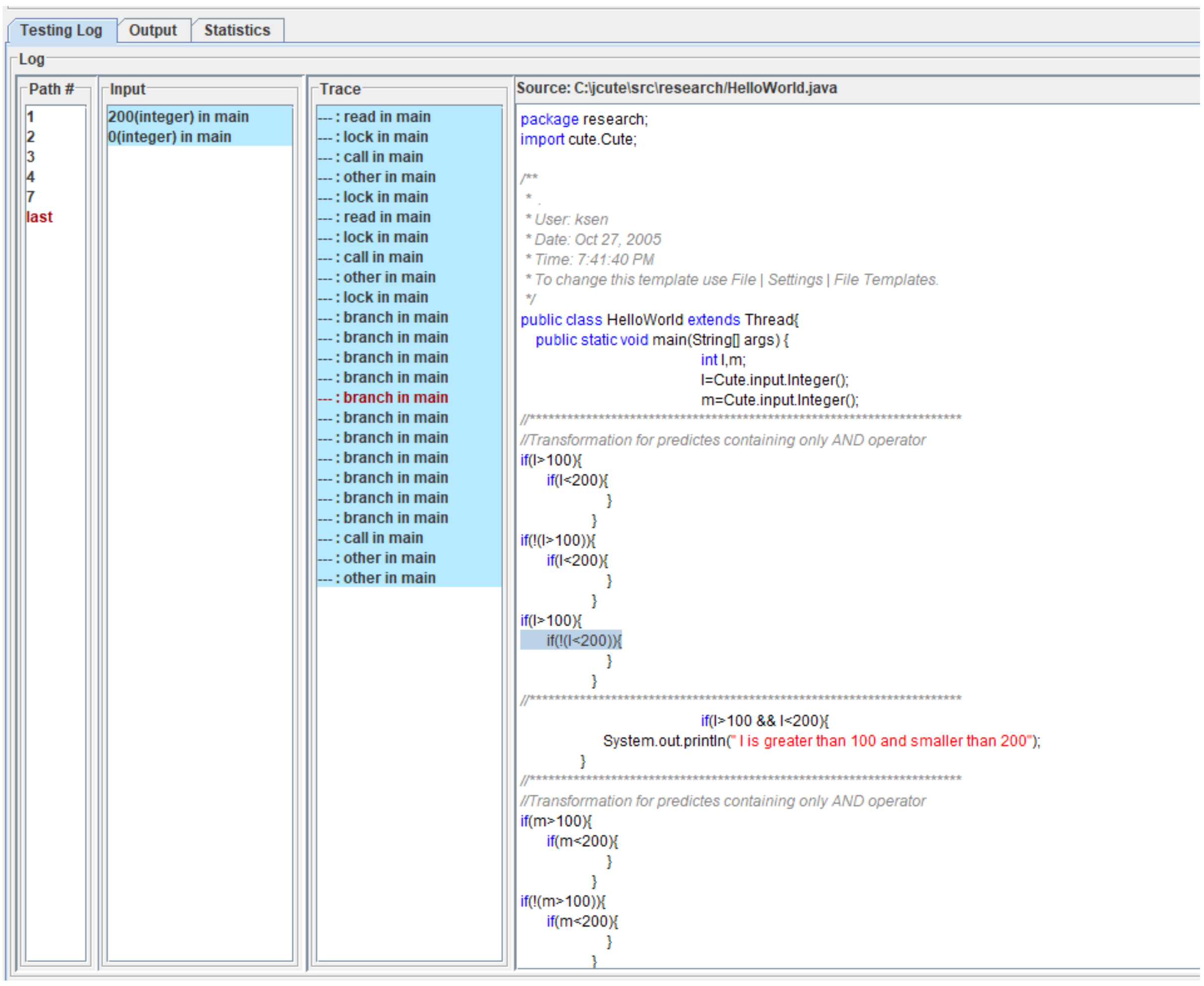
Other Options:



Testing Log | Output | Statistics

Output

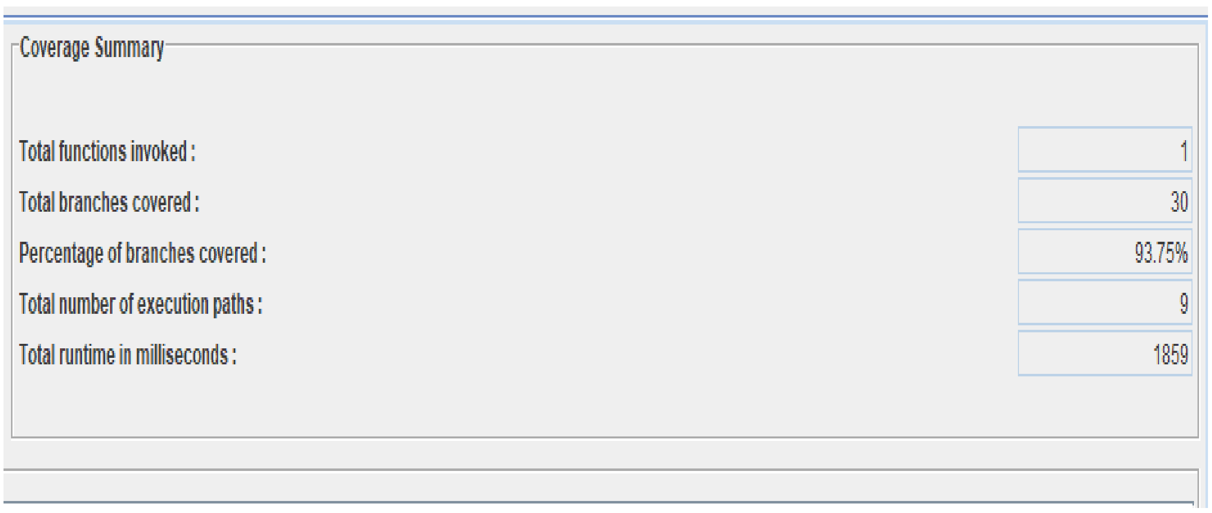
```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j:-r: cute.RunOnce research.HelloWorld.main
 l is greater than 100 and smaller than 200
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.HelloWorld.main
 l is greater than 100 and smaller than 200
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j:-r: cute.RunOnce research.HelloWorld.main
 l is greater than 100 and smaller than 200
 m is greater than 100 and smaller than 200
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j:-r: cute.RunOnce research.HelloWorld.main
 l is greater than 100 and smaller than 200
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j:-r: cute.RunOnce research.HelloWorld.main
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.HelloWorld.main
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j:-r: cute.RunOnce research.HelloWorld.main
 m is greater than 100 and smaller than 200
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j:-r: cute.RunOnce research.HelloWorld.main
**************************** One complete search is over *************************
Exit 2

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.HelloWorld.main
Exit 0
```



```
Path #    Input                    Trace
1         200(integer) in main     --- : read in main
2         0(integer) in main       --- : lock in main
3                                  --- : call in main
4                                  --- : other in main
7                                  --- : lock in main
last                               --- : read in main
                                   --- : lock in main
                                   --- : call in main
                                   --- : other in main
                                   --- : lock in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : branch in main
                                   --- : call in main
                                   --- : other in main
                                   --- : other in main
```

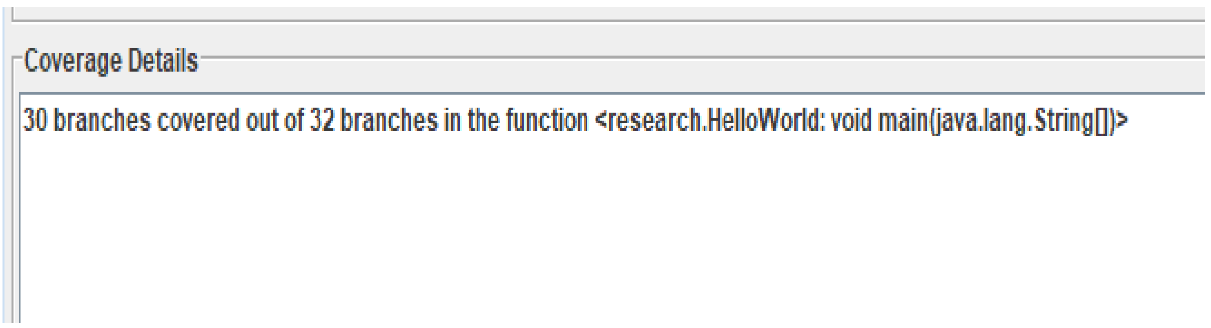
Source: C:\jcute\src\research/HelloWorld.java

```
package research;
import cute.Cute;

/**
 * .
 * User: ksen
 * Date: Oct 27, 2005
 * Time: 7:41:40 PM
 * To change this template use File | Settings | File Templates.
 */
public class HelloWorld extends Thread{
   public static void main(String[] args) {
                              int l,m;
                              l=Cute.input.Integer();
                              m=Cute.input.Integer();
//*********************************************************************
//Transformation for predictes containing only AND operator
if(l>100){
     if(l<200){
              }
           }
if(!(l>100)){
     if(l<200){
              }
           }
if(l>100){
     if(!(l<200)){
              }
           }
//*********************************************************************
                              if(l>100 && l<200){
               System.out.println(" I is greater than 100 and smaller than 200");
          }
//*********************************************************************
//Transformation for predictes containing only AND operator
if(m>100){
     if(m<200){
              }
           }
if(!(m>100)){
     if(m<200){
              }
           }
```



Coverage Summary

Total functions invoked :	1
Total branches covered :	30
Percentage of branches covered :	93.75%
Total number of execution paths :	9
Total runtime in milliseconds :	1859



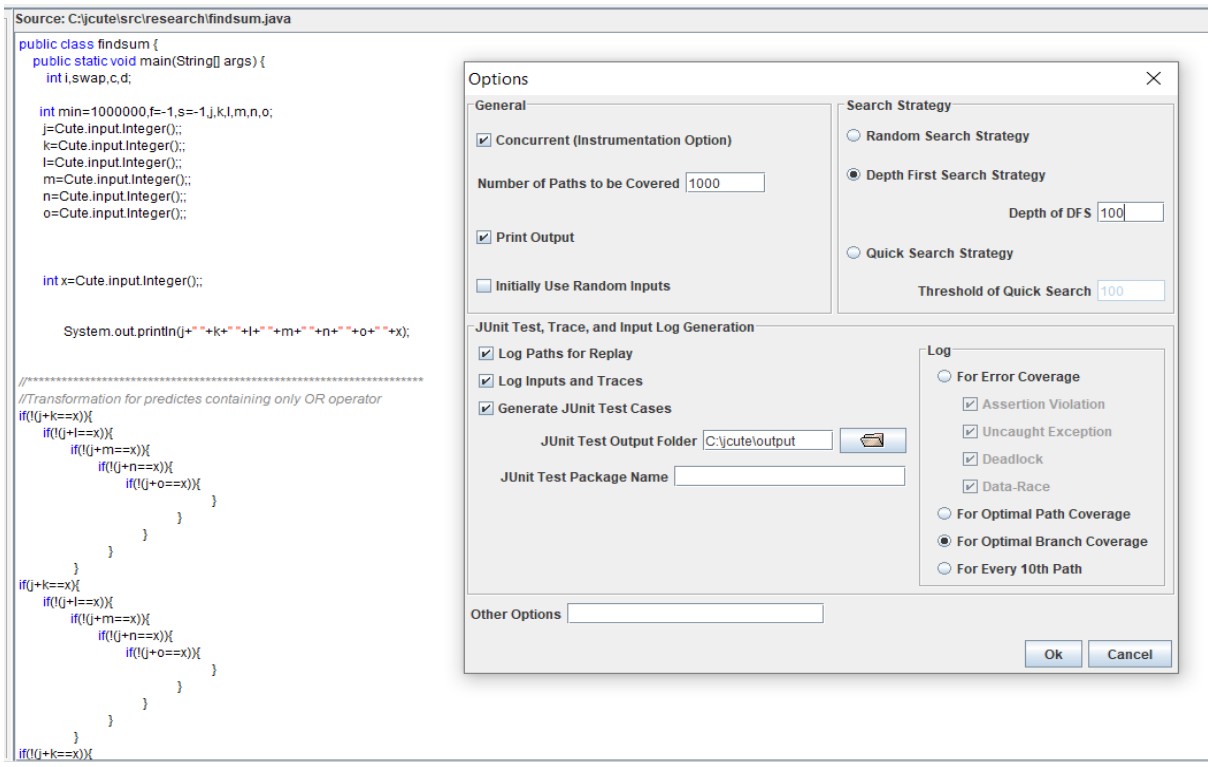
Coverage Details

30 branches covered out of 32 branches in the function <research.HelloWorld: void main(java.lang.String[])>

```java
/* JUnit test case generated automatically by CUTE */
import junit.framework.*;

public class research_Largest_main_Test extends TestCase
implements cute.Input {
    private Object[] input;
    private int i;

    public research_Largest_main_Test(String name){
        super(name);
    }

    public boolean Boolean() {
        return ((Boolean)input[i++]).booleanValue();
    }

    public short Short() {
        return ((Short)input[i++]).shortValue();
    }

    public int Integer() {
        return ((Integer)input[i++]).intValue();
    }

    public long Long() {
        return ((Long)input[i++]).longValue();
    }

    public float Float() {
        return ((Float)input[i++]).floatValue();
    }

    public double Double() {
        return ((Double)input[i++]).doubleValue();
    }

    public char Character() {
        return ((Character)input[i++]).charValue();
    }

    public byte Byte() {
        return ((Byte)input[i++]).byteValue();
    }

    public Object Object(String type) {
        return input[i++];
    }

    public void test1(){
        i=0;
        input = new Object[3];
        input[i++] = new Integer(-1587573543);
        input[i++] = new Integer(495673980);
        input[i++] = new Integer(-611698592);
        i=0;
```

```java
        cute.Cute.input = this;
        research.Largest.main(null);
    }

    public void test2(){
        i=0;
        input = new Object[3];
        input[i++] = new Integer(0);
        input[i++] = new Integer(1);
        input[i++] = new Integer(0);
        i=0;
        cute.Cute.input = this;
        research.Largest.main(null);
    }

    public void test3(){
        i=0;
        input = new Object[3];
        input[i++] = new Integer(0);
        input[i++] = new Integer(1);
        input[i++] = new Integer(1);
        i=0;
        cute.Cute.input = this;
        research.Largest.main(null);
    }

    public void test4(){
        i=0;
        input = new Object[3];
        input[i++] = new Integer(0);
        input[i++] = new Integer(1);
        input[i++] = new Integer(2);
        i=0;
        cute.Cute.input = this;
        research.Largest.main(null);
    }
```

4) findsum.java



Source: C:\jcute\src\research\findsum.java

```
public class findsum {
   public static void main(String[] args) {
      int i,swap,c,d;

     int min=1000000,f=-1,s=-1,j,k,l,m,n,o;
      j=Cute.input.Integer();;
      k=Cute.input.Integer();;
      l=Cute.input.Integer();;
      m=Cute.input.Integer();;
      n=Cute.input.Integer();;
      o=Cute.input.Integer();;



      int x=Cute.input.Integer();;


         System.out.println(j+" "+k+" "+l+" "+m+" "+n+" "+o+" "+x);


//***********************************************************************
//Transformation for predictes containing only OR operator
if(!(j+k==x)){
      if(!(j+l==x)){
            if(!(j+m==x)){
                  if(!(j+n==x)){
                        if(!(j+o==x)){
                                    }
                              }
                        }
                  }
            }
if(j+k==x){
      if(!(j+l==x)){
            if(!(j+m==x)){
                  if(!(j+n==x)){
                        if(!(j+o==x)){
                                    }
                              }
                        }
                  }
            }
if(!(j+k==x)){
```

Options

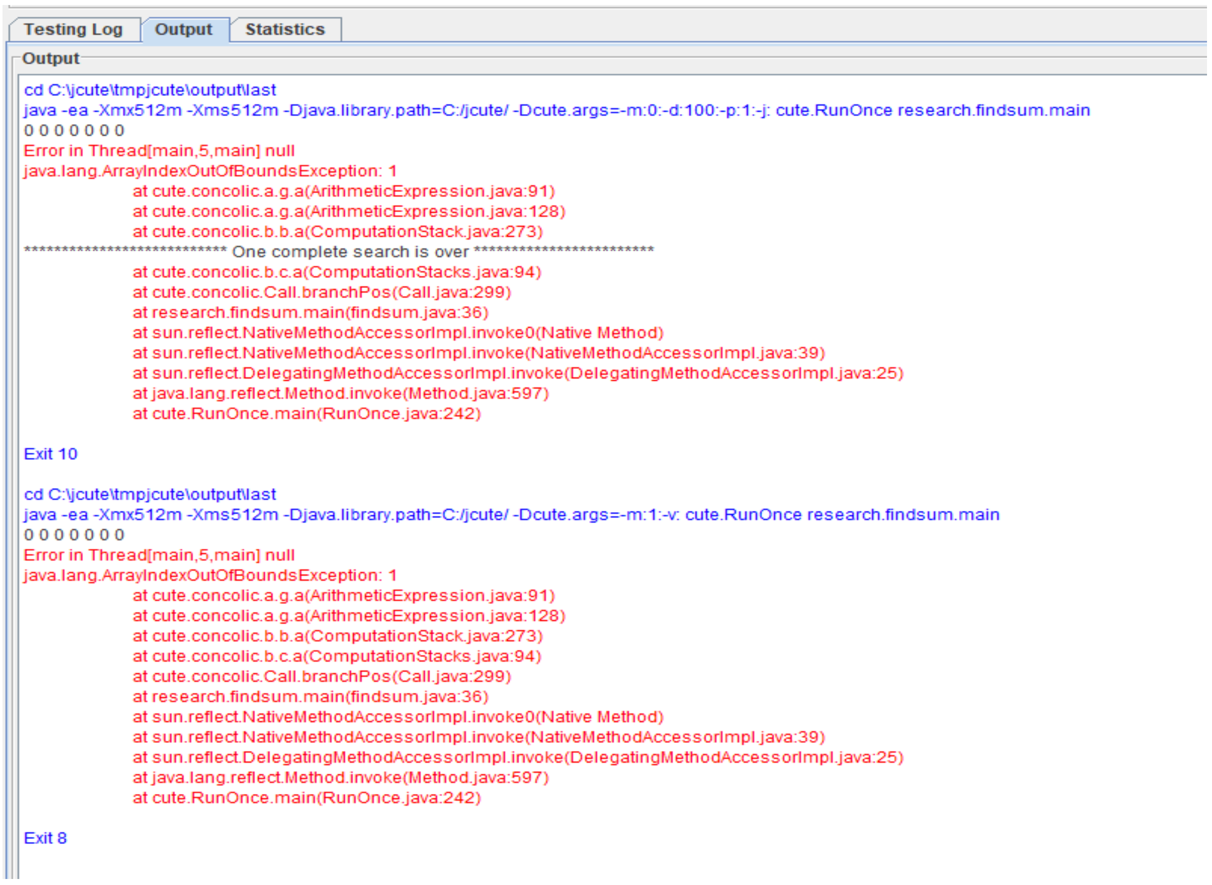
General
- [✓] Concurrent (Instrumentation Option)
- Number of Paths to be Covered: 1000
- [✓] Print Output
- [ ] Initially Use Random Inputs

Search Strategy
- ○ Random Search Strategy
- ● Depth First Search Strategy
  - Depth of DFS: 100
- ○ Quick Search Strategy
  - Threshold of Quick Search: 100

JUnit Test, Trace, and Input Log Generation
- [✓] Log Paths for Replay
- [✓] Log Inputs and Traces
- [✓] Generate JUnit Test Cases
- JUnit Test Output Folder: C:\jcute\output
- JUnit Test Package Name:

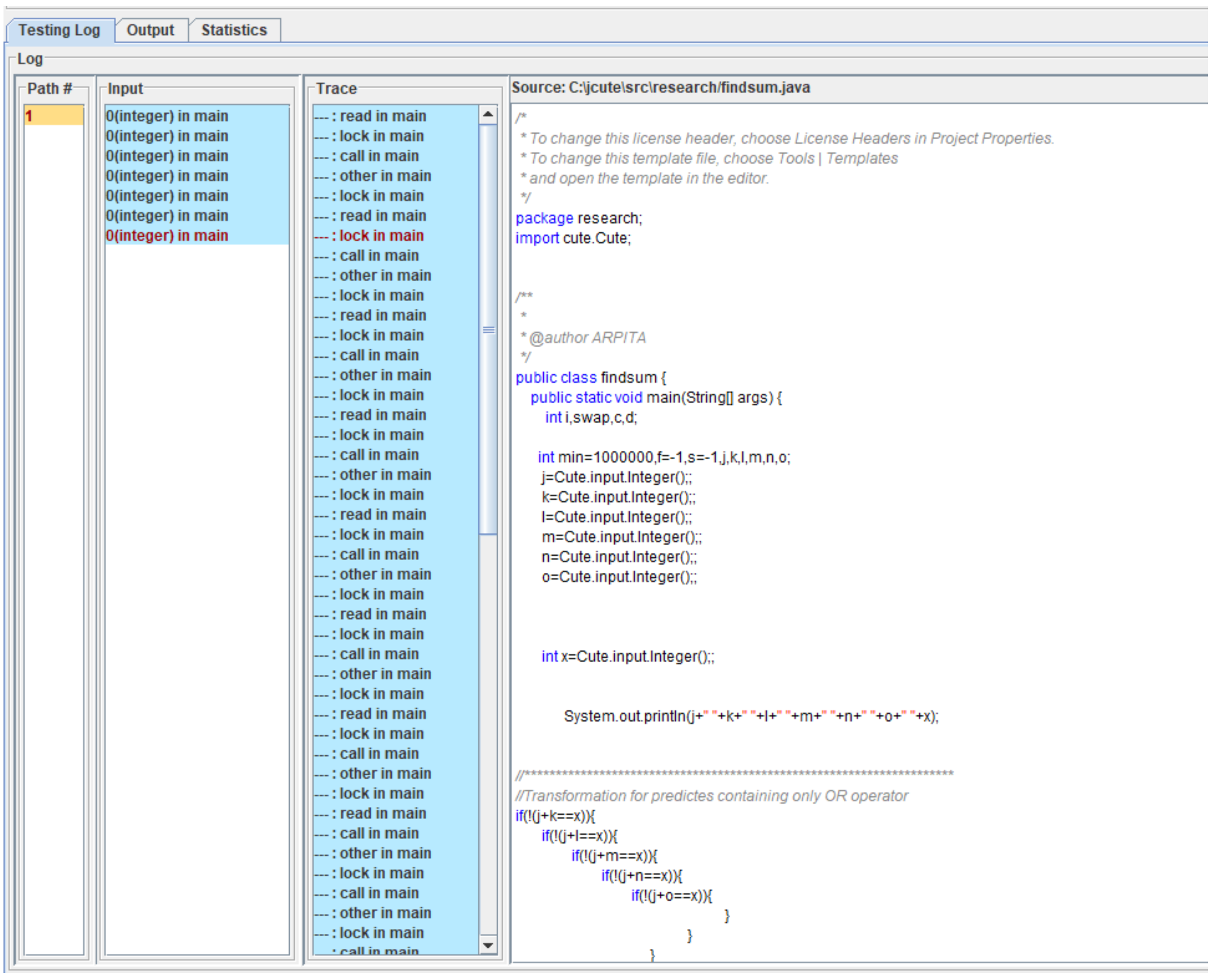
Log
- ○ For Error Coverage
  - [✓] Assertion Violation
  - [✓] Uncaught Exception
  - [✓] Deadlock
  - [✓] Data-Race
- ○ For Optimal Path Coverage
- ● For Optimal Branch Coverage
- ○ For Every 10th Path

Other Options:



Output

```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:100:-p:1:-j: cute.RunOnce research.findsum.main
0 0 0 0 0 0 0
Error in Thread[main,5,main] null
java.lang.ArrayIndexOutOfBoundsException: 1
                at cute.concolic.a.g.a(ArithmeticExpression.java:91)
                at cute.concolic.a.g.a(ArithmeticExpression.java:128)
                at cute.concolic.b.b.a(ComputationStack.java:273)
*************************** One complete search is over ************************
                at cute.concolic.b.c.a(ComputationStacks.java:94)
                at cute.concolic.Call.branchPos(Call.java:299)
                at research.findsum.main(findsum.java:36)
                at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
                at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
                at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
                at java.lang.reflect.Method.invoke(Method.java:597)
                at cute.RunOnce.main(RunOnce.java:242)

Exit 10

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce research.findsum.main
0 0 0 0 0 0 0
Error in Thread[main,5,main] null
java.lang.ArrayIndexOutOfBoundsException: 1
                at cute.concolic.a.g.a(ArithmeticExpression.java:91)
                at cute.concolic.a.g.a(ArithmeticExpression.java:128)
                at cute.concolic.b.b.a(ComputationStack.java:273)
                at cute.concolic.b.c.a(ComputationStacks.java:94)
                at cute.concolic.Call.branchPos(Call.java:299)
                at research.findsum.main(findsum.java:36)
                at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
                at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
                at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
                at java.lang.reflect.Method.invoke(Method.java:597)
                at cute.RunOnce.main(RunOnce.java:242)

Exit 8
```



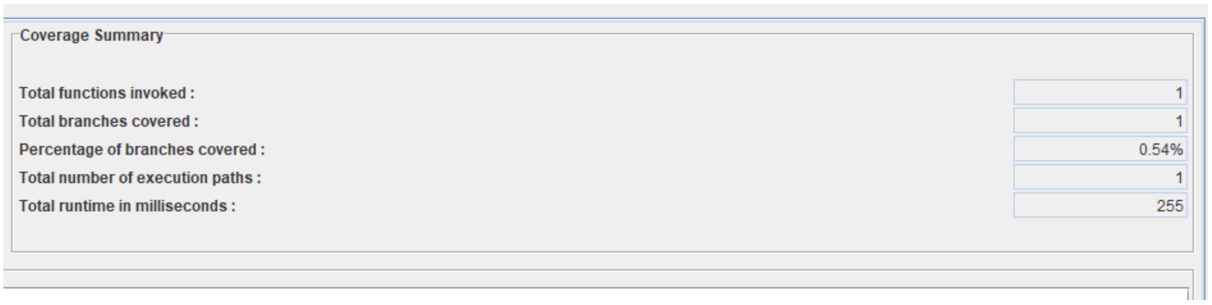
Path #: 1

```
0(integer) in main
0(integer) in main
0(integer) in main
0(integer) in main
0(integer) in main
0(integer) in main
0(integer) in main
```

```
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
```

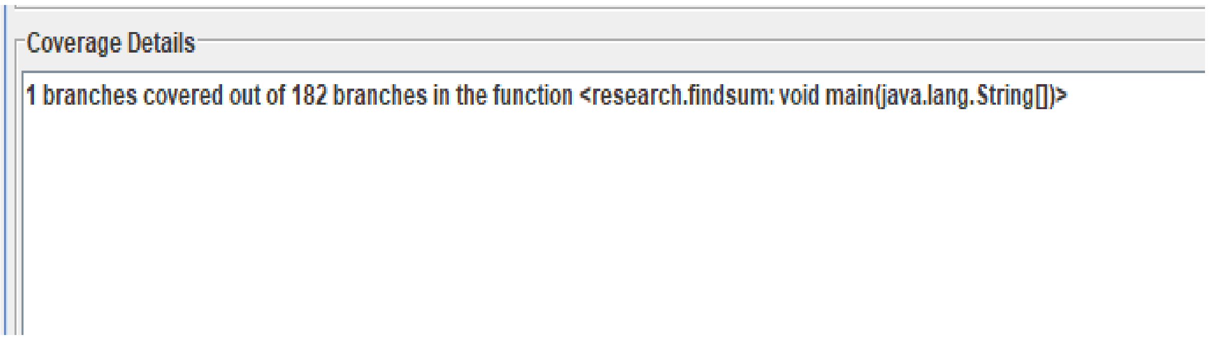
Source: C:\jcute\src\research/findsum.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package research;
import cute.Cute;


/**
 *
 * @author ARPITA
 */
public class findsum {
    public static void main(String[] args) {
       int i,swap,c,d;

      int min=1000000,f=-1,s=-1,j,k,l,m,n,o;
       j=Cute.input.Integer();;
       k=Cute.input.Integer();;
       l=Cute.input.Integer();;
       m=Cute.input.Integer();;
       n=Cute.input.Integer();;
       o=Cute.input.Integer();;



       int x=Cute.input.Integer();;


          System.out.println(j+" "+k+" "+l+" "+m+" "+n+" "+o+" "+x);

//*******************************************************************
//Transformation for predictes containing only OR operator
if(!(j+k==x)){
       if(!(j+l==x)){
              if(!(j+m==x)){
                     if(!(j+n==x)){
                            if(!(j+o==x)){
                                          }
                                   }
                            }
```



Coverage Summary

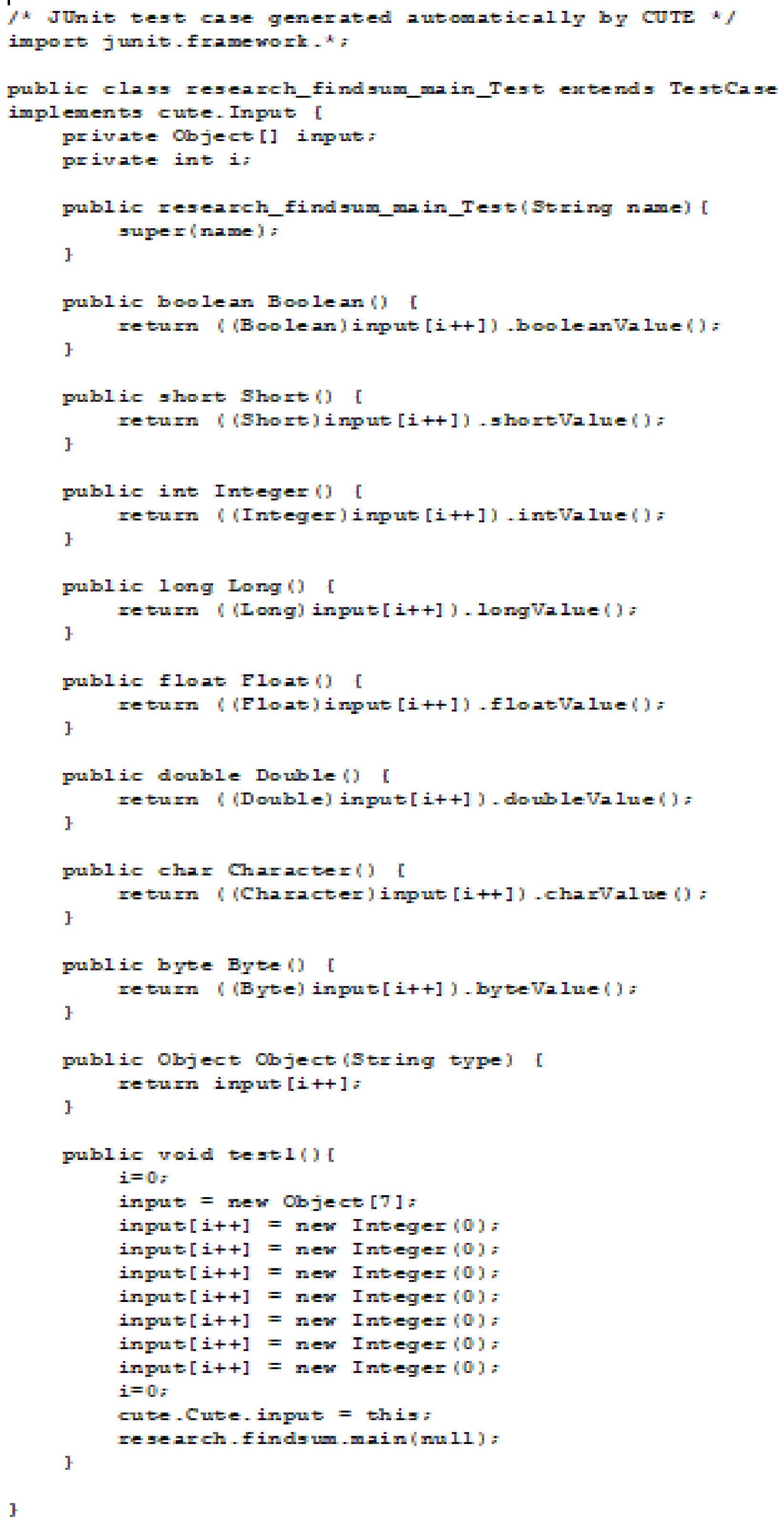
Total functions invoked :	1
Total branches covered :	1
Percentage of branches covered :	0.54%
Total number of execution paths :	1
Total runtime in milliseconds :	255



Coverage Details

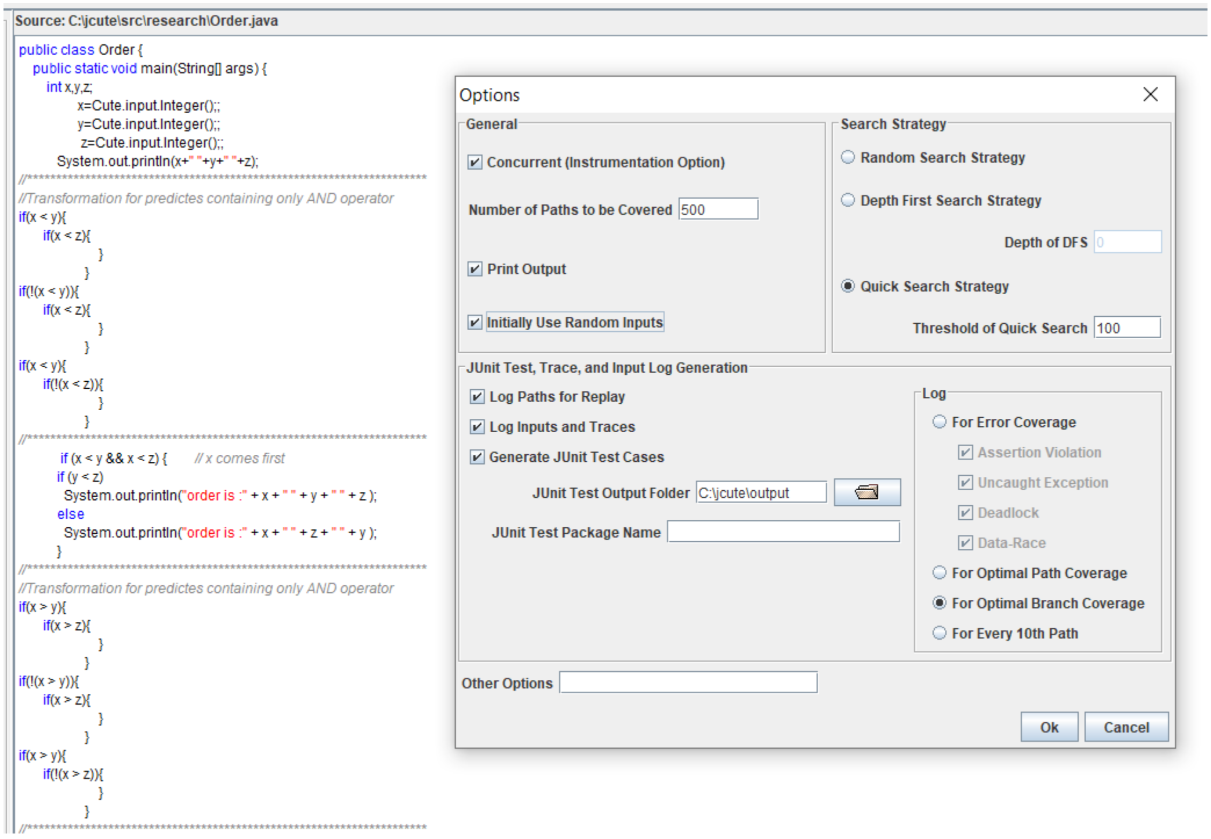
```
1 branches covered out of 182 branches in the function <research.findsum: void main(java.lang.String[])>
```

Test case file generated : -



```java
/* JUnit test case generated automatically by CUTE */
import junit.framework.*;

public class research_findsum_main_Test extends TestCase
implements cute.Input {
    private Object[] input;
    private int i;

    public research_findsum_main_Test(String name){
        super(name);
    }

    public boolean Boolean() {
        return ((Boolean)input[i++]).booleanValue();
    }

    public short Short() {
        return ((Short)input[i++]).shortValue();
    }

    public int Integer() {
        return ((Integer)input[i++]).intValue();
    }

    public long Long() {
        return ((Long)input[i++]).longValue();
    }

    public float Float() {
        return ((Float)input[i++]).floatValue();
    }

    public double Double() {
        return ((Double)input[i++]).doubleValue();
    }

    public char Character() {
        return ((Character)input[i++]).charValue();
    }

    public byte Byte() {
        return ((Byte)input[i++]).byteValue();
    }

    public Object Object(String type) {
        return input[i++];
    }

    public void test1(){
        i=0;
        input = new Object[7];
        input[i++] = new Integer(0);
        input[i++] = new Integer(0);
        input[i++] = new Integer(0);
        input[i++] = new Integer(0);
        input[i++] = new Integer(0);
        input[i++] = new Integer(0);
        input[i++] = new Integer(0);
        i=0;
        cute.Cute.input = this;
        research.findsum.main(null);
    }

}
```

5) order.java



Source: C:\jcute\src\research\Order.java

```
public class Order {
   public static void main(String[] args) {
      int x,y,z;
              x=Cute.input.Integer();;
              y=Cute.input.Integer();;
               z=Cute.input.Integer();;
        System.out.println(x+" "+y+" "+z);
//********************************************************************
//Transformation for predictes containing only AND operator
if(x < y){
     if(x < z){
                }
             }
if(!(x < y)){
     if(x < z){
                }
             }
if(x < y){
     if(!(x < z)){
                }
             }
//********************************************************************
          if (x < y && x < z) {      // x comes first
         if (y < z)
          System.out.println("order is :" + x + " " + y + " " + z );
         else
          System.out.println("order is :" + x + " " + z + " " + y );
         }
//********************************************************************
//Transformation for predictes containing only AND operator
if(x > y){
     if(x > z){
                }
             }
if(!(x > y)){
     if(x > z){
                }
             }
if(x > y){
     if(!(x > z)){
                }
             }
//********************************************************************
```

Options

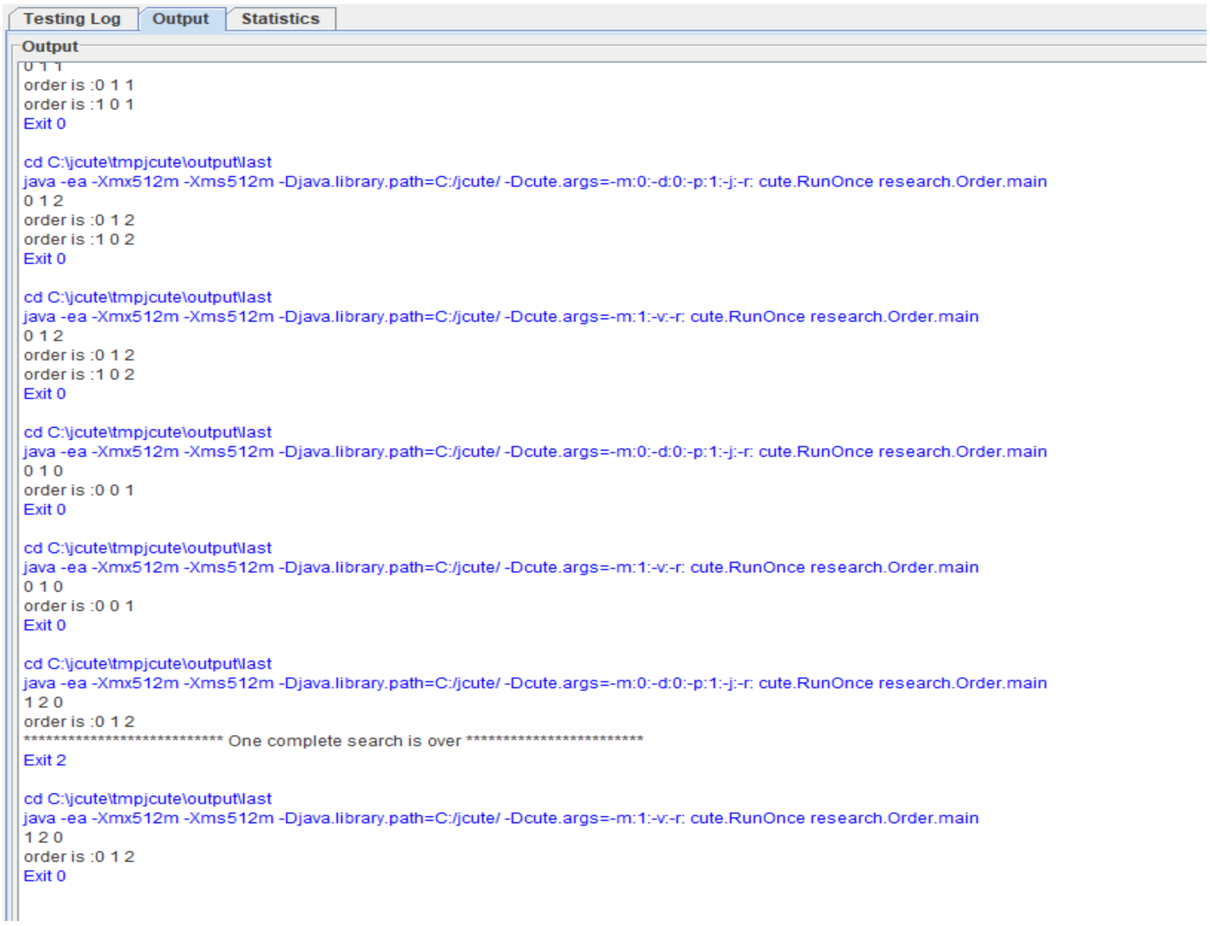
General
- [✓] Concurrent (Instrumentation Option)
- Number of Paths to be Covered: 500
- [✓] Print Output
- [✓] Initially Use Random Inputs

Search Strategy
- ○ Random Search Strategy
- ○ Depth First Search Strategy
- Depth of DFS: 0
- ● Quick Search Strategy
- Threshold of Quick Search: 100

JUnit Test, Trace, and Input Log Generation
- [✓] Log Paths for Replay
- [✓] Log Inputs and Traces
- [✓] Generate JUnit Test Cases
- JUnit Test Output Folder: C:\jcute\output
- JUnit Test Package Name:

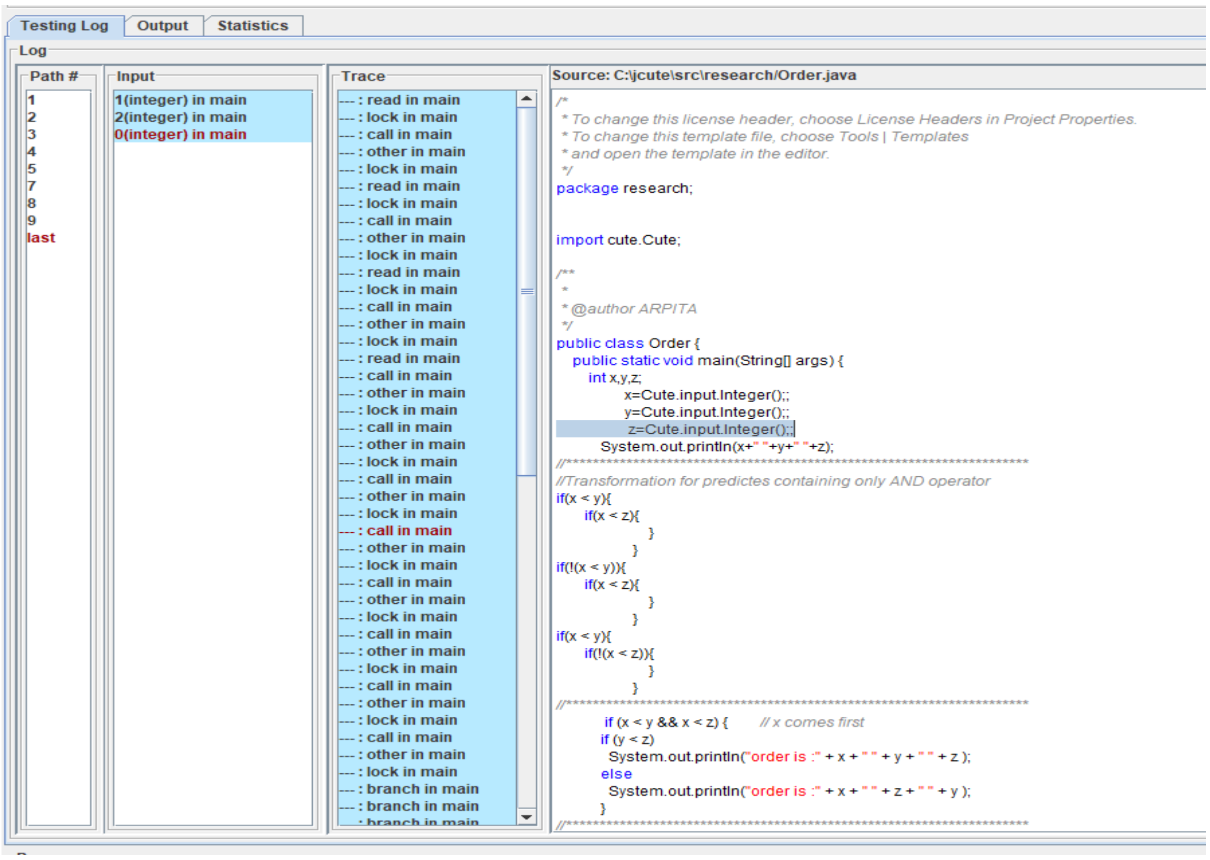
Log
- ○ For Error Coverage
  - [✓] Assertion Violation
  - [✓] Uncaught Exception
  - [✓] Deadlock
  - [✓] Data-Race
- ○ For Optimal Path Coverage
- ● For Optimal Branch Coverage
- ○ For Every 10th Path

Other Options:



Output

```
0 1 1
order is :0 1 1
order is :1 0 1
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Order.main
0 1 2
order is :0 1 2
order is :1 0 2
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.Order.main
0 1 2
order is :0 1 2
order is :1 0 2
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Order.main
0 1 0
order is :0 0 1
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.Order.main
0 1 0
order is :0 0 1
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j:-r: cute.RunOnce research.Order.main
1 2 0
order is :0 1 2
**************************** One complete search is over *************************
Exit 2

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v:-r: cute.RunOnce research.Order.main
1 2 0
order is :0 1 2
Exit 0
```



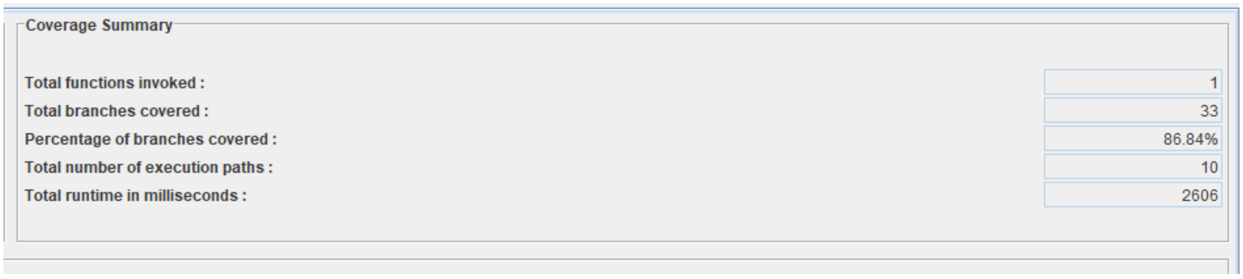
```
1
2
3
4
5
7
8
9
last
```

```
1(integer) in main
2(integer) in main
0(integer) in main
```

```
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : read in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : call in main
--- : other in main
--- : lock in main
--- : branch in main
--- : branch in main
--- : branch in main
```

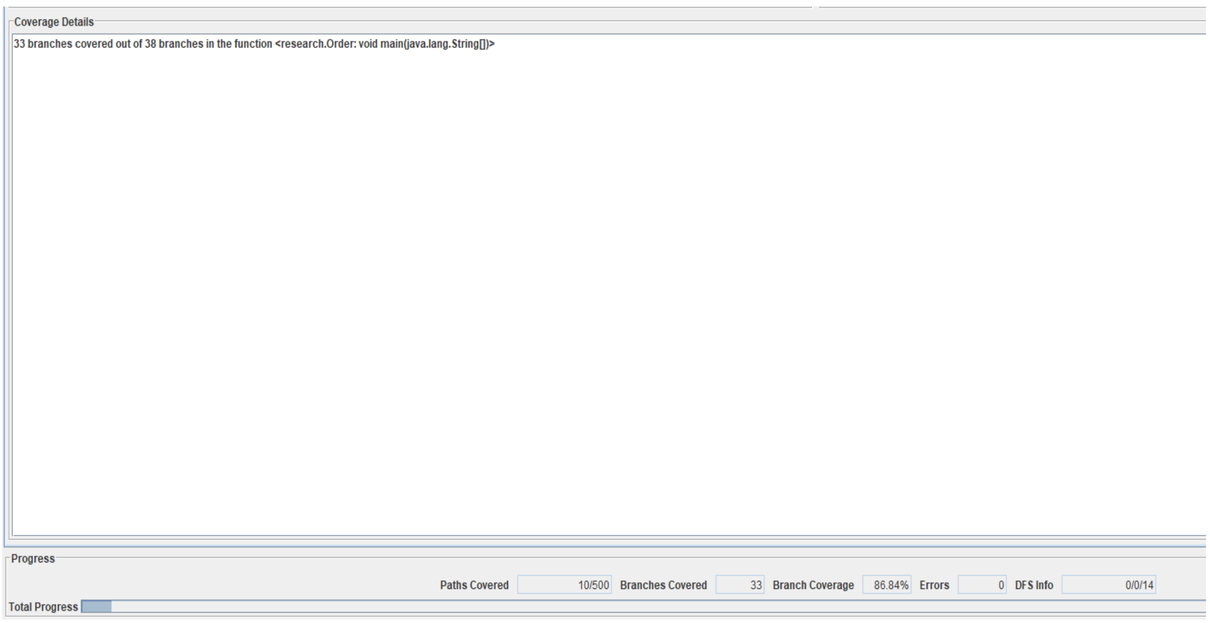
Source: C:\jcute\src\research/Order.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package research;


import cute.Cute;

/**
 *
 * @author ARPITA
 */
public class Order {
   public static void main(String[] args) {
      int x,y,z;
             x=Cute.input.Integer();;
             y=Cute.input.Integer();;
              z=Cute.input.Integer();;
         System.out.println(x+" "+y+" "+z);
//*********************************************************************
//Transformation for predictes containing only AND operator
if(x < y){
     if(x < z){
                }
             }
if(!(x < y)){
     if(x < z){
                }
             }
if(x < y){
     if(!(x < z)){
                }
             }
//*********************************************************************
          if (x < y && x < z) {       // x comes first
         if (y < z)
          System.out.println("order is :" + x + " " + y + " " + z );
         else
          System.out.println("order is :" + x + " " + z + " " + y );
         }
//*********************************************************************
```



Coverage Summary

Total functions invoked :	1
Total branches covered :	33
Percentage of branches covered :	86.84%
Total number of execution paths :	10
Total runtime in milliseconds :	2606



Coverage Details

```
33 branches covered out of 38 branches in the function <research.Order: void main(java.lang.String[])>
```

Progress

Paths Covered	Branches Covered	Branch Coverage	Errors	DFS Info
10/500	33	86.84%	0	0/0/14

Total Progress

```
/* JUnit test case generated automatically by CUTE */
import junit.framework.*;

public class research_Order_main_Test extends TestCase implements
cute.Input {
    private Object[] input;
    private int i;

    public research_Order_main_Test(String name){
        super(name);
    }

    public boolean Boolean() {
        return ((Boolean)input[i++]).booleanValue();
    }

    public short Short() {
        return ((Short)input[i++]).shortValue();
    }

    public int Integer() {
        return ((Integer)input[i++]).intValue();
    }

    public long Long() {
        return ((Long)input[i++]).longValue();
    }

    public float Float() {
        return ((Float)input[i++]).floatValue();
    }

    public double Double() {
        return ((Double)input[i++]).doubleValue();
    }

    public char Character() {
        return ((Character)input[i++]).charValue();
    }

    public byte Byte() {
        return ((Byte)input[i++]).byteValue();
    }

    public Object Object(String type) {
        return input[i++];
    }

    public void test1(){
        i=0;
        input = new Object[3];
        input[i++] = new Integer(187708191);
        input[i++] = new Integer(-874837839);
        input[i++] = new Integer(219816130);
        i=0;
        cute.Cute.input = this;
        research.Order.main(null);
    }

    public void test2(){
        i=0;
```

```
}

public void test3(){
    i=0;
    input = new Object[3];
    input[i++] = new Integer(0);
    input[i++] = new Integer(0);
    input[i++] = new Integer(0);
    i=0;
    cute.Cute.input = this;
    research.Order.main(null);
}

public void test4(){
    i=0;
    input = new Object[3];
    input[i++] = new Integer(1);
    input[i++] = new Integer(1);
    input[i++] = new Integer(0);
    i=0;
    cute.Cute.input = this;
    research.Order.main(null);
}

public void test5(){
    i=0;
    input = new Object[3];
    input[i++] = new Integer(1);
    input[i++] = new Integer(0);
    input[i++] = new Integer(0);
    i=0;
    cute.Cute.input = this;
    research.Order.main(null);
}

public void test7(){
    i=0;
    input = new Object[3];
    input[i++] = new Integer(0);
    input[i++] = new Integer(1);
    input[i++] = new Integer(1);
    i=0;
    cute.Cute.input = this;
    research.Order.main(null);
}

public void test8(){
    i=0;
    input = new Object[3];
    input[i++] = new Integer(0);
    input[i++] = new Integer(1);
    input[i++] = new Integer(2);
    i=0;
    cute.Cute.input = this;
    research.Order.main(null);
}

public void test9(){
    i=0;
    input = new Object[3];
    input[i++] = new Integer(0);
    input[i++] = new Integer(1);
    input[i++] = new Integer(0);
```