Software Testing, Lab 3, March 14, 2023.

1 Tasks:

1.Install Major Mutation Framework. The instruction of how to install and use Major can be seen in https://mutation-testing.org/tutorial.html

2.Coding a program named 'UpgradedTriangle'. Given the length value (integer) of 3 sides of a triangle. Finish 2 functions respectively,

a) classifying the triangle

given 3 length of sides(integers), output the shape of triangle made up by given sides. (Output a String, the shape could be "SCALENE","EQUILATERAL","ISOSCELES","INVALID".)

b) calculating the area of valid triangle.

given 3 length of sides(integers), if these 3 sides can make up a valid triangle, output the area of the triangle (double or float), otherwise, return 0. (reference : Heron's formula)

3.Write testing cases for 2 functions with Junit according to your previous study (MC/DC, boundary value, equivalence partitioning, etc.), guarantee the sufficiency and diversity of your test set. Each function should have 15 test cases.

4.Then run mutants on the test sets with Major Mutation Framework. In order for you to learn how to modify the build.xml file, please try to make the final file structure like follow:

```
├── Lab3
|   ├── src
|   |   ├── main
|   |   |   ├── java
|   |   |   |   └── UpgradedTriangle.java //source code
|   |   ├── test
|   |   |   ├── java
|   |   |   |   └── UpgradedTriangleTest.java //junit code
|   ├── target
|   |   ├── UpgradedTriangle.class
|   |   └── UpgradedTriangleTest.class
|   ├── build.xml
|   ├── covMap.csv
|   ├── details.csv
|   ├── major.log
|   ├── mutants.log
|   ├── preprocessing.ser
|   ├── summary.csv
└── └── testMap.csv
```

5. Analyzing the report provided by Major. Calculate these values:

- The number of mutants generated
- The number of mutants covered by the test suite
- The number of mutants killed by the test suite
- The number of live mutants
- The overall mutation score / adequacy of the test suite

Discuss and explain your results: (Here are some Viewpoints you could discuss)

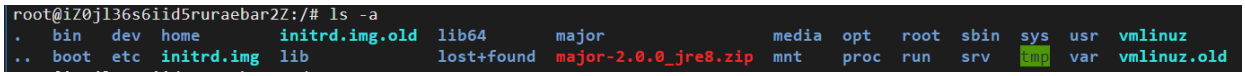
- What do the results tell you about your test suite?
- Does the test suite exhibit weaknesses? How can it be improved?
- Does the test suite exhibit strengths? How do you recognize them?
- Do you have any other interesting insights or opinions on the experience?
- Among the generated mutants, If both killed and unkilled mutants were generated, what was the type of operator used? How was it applied to the code (how did the code change)?
- According to your mutation analysis result, which part of the source code need to be strengthened in further coding? Which test case in your suite are more important compared with others.

2 配置major

1、linux下安装jdk，根据提示命令行安装openjdk-8。

2、配置major

(1) 在linux环境下，将major压缩包解压，放在主目录下，如图:



(2) 配置环境变量

命令行输入以下命令。

```
sudo vim /etc/profile
```

在文件末尾加上以下命令。

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_321
export MAJOR_HOME=/major
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$PATH:$JAVA_HOME/bin
export PATH=${MAJOR_HOME}/bin:$PATH
```

命令行输入以下命令，使环境生效。

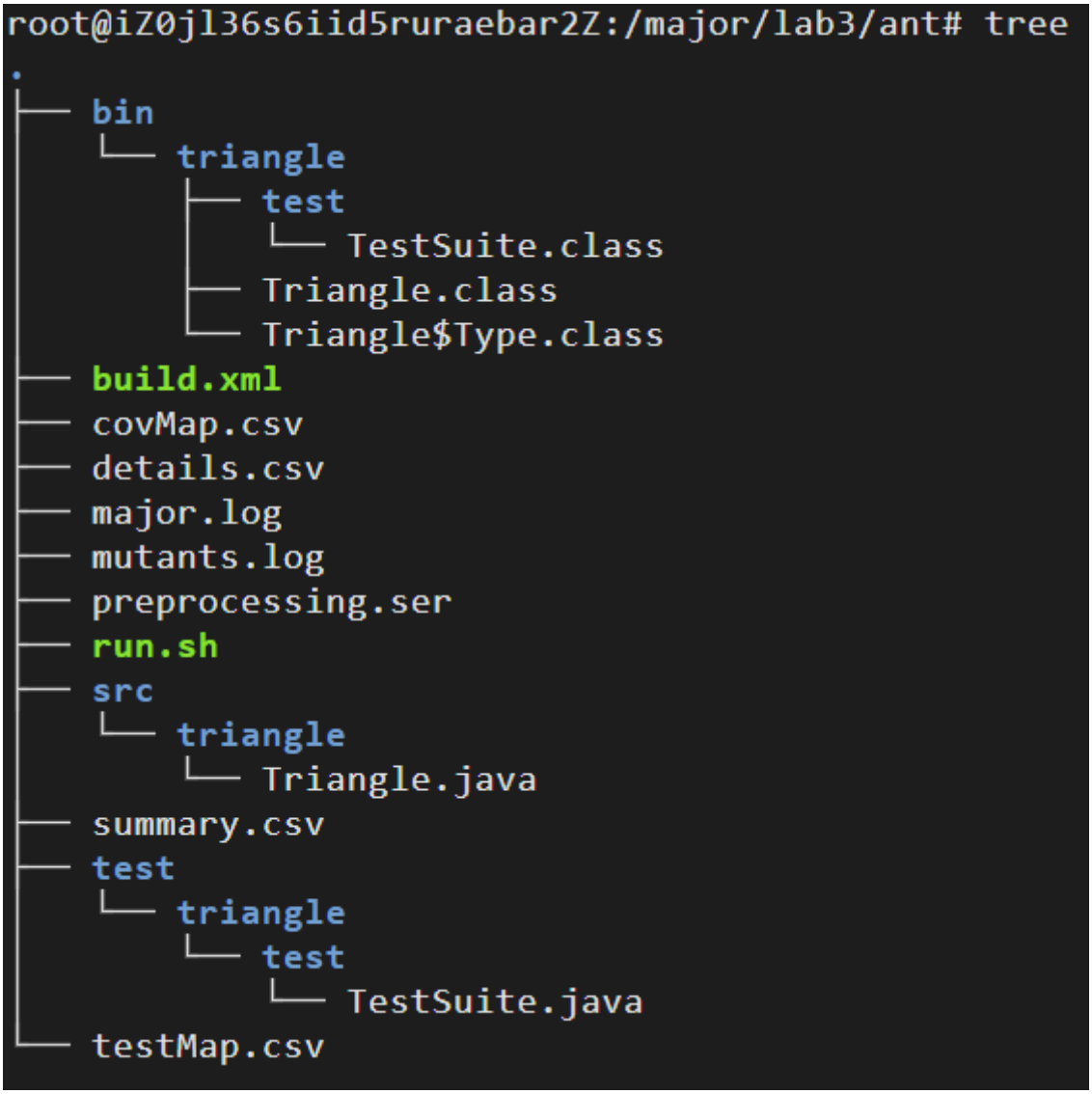
```
source /etc/profile
```

验证结果如下：

```
root@iZ0jl36s6iid5ruraebar2Z:/major# javac -version
javac 1.8.0_362
root@iZ0jl36s6iid5ruraebar2Z:/major# ant -version
Apache Ant(TM) version 1.8.4-Major-v2.0.0 compiled on January 22 2023
```

3 代码编写

仿照example中的文件组织结构，在major目录下构建lab3/ant。



变异体生成的指令配置选项使用的是tutorial.mml.bin，实际上这个是专门为example/ant下的Triangle.java使用的变异体测试配置，这里我们需要进行修改，我们是对UpgradedTriangle.java中的所有方法都进行变异体测试，同时选择所有的变异算子，因此我们应该将-DumpOp后面的配置项改为ALL。

```
echo
echo "Compiling and mutating project"
echo "(ant -Dmutator=\"=mml:\$MAJOR_HOME/mml/all.mml.bin\" clean compile)"
echo
$MAJOR_HOME/bin/ant -Dmutator="mml:$MAJOR_HOME/mml/all.mml.bin" clean compile
```

build.xml与example中的类似。具体如下：

```
<project name="Triangle" default="compile" basedir=".">

  <!-- ###############################################
       Path to major.jar and (optional) mml file
       Use -Dmutator="mml:<path-to-mml-file>" to set a
       particular (compiled) mml-file.
      ############################################### -->
  <property name="major.jar" value="../../lib/major.jar" />
  <property name="mutator" value="" />
  <property name="plugin" value="-Xplugin:MajorPlugin ${mutator}" />


  <!-- Target to clean up -->
  <target name="clean" description="Clean">
    <delete dir="bin" />
    <delete>
      <fileset dir="." includes="*.csv" />
      <fileset dir="." includes="*.log" />
    </delete>
  </target>

  <!-- Target to initialize build -->
  <target name="init">
    <mkdir dir="bin" />
  </target>

  <!-- Target to compile the project -->
  <target name="compile" depends="init" description="Compile">
    <javac includeantruntime="true"
      srcdir="src"
      destdir="bin"
      debug="yes">

      <classpath location="${major.jar}" />

      <compilerarg value="${plugin}" />
    </javac>
  </target>

  <!-- Target to compile the test suite -->
  <target name="compile.tests" depends="compile" description="Compile all tests">
    <javac includeantruntime="true"
      srcdir="test"
      destdir="bin"
      debug="yes">
    </javac>
  </target>

  <!-- The adapted mutation test target -->
```

```
  <target name="mutation.test" description="Run mutation analysis for all unit test
cases">
    <echo message="Running mutation analysis ..." />
    <junit printsummary="false"
      showoutput="false"
      mutationAnalysis="true"
      summaryFile="summary.csv"
      mutantDetailsFile="details.csv">

      <classpath path="bin" />
      <batchtest fork="false">
        <fileset dir="test">
          <include name="**/*Test*.java" />
        </fileset>
      </batchtest>
    </junit>
  </target>

  <!-- The original test target -->
  <target name="test" depends="compile.tests" description="Run all unit test cases">
    <echo message="Running unit tests ..." />
    <junit printsummary="true"
      showoutput="true"
      haltonfailure="false">
      <formatter type="plain" usefile="false" />
      <classpath path="bin" />

      <batchtest fork="no">
        <fileset dir="test">
          <include name="**/*Test*.java" />
        </fileset>
      </batchtest>
    </junit>
  </target>
</project>
```

Triangle.java 代码如下

```
public class Triangle {
  public static enum Type {
    INVALID, SCALENE, EQUILATERAL, ISOSCELES
  };

  public static Type classify(int a, int b, int c) {
    if (a <= 0 || b <= 0 || c <= 0) {
      return Type.INVALID;
    }
    if (a + b <= c || a + c <= b || b + c <= a) {
      return Type.INVALID;
    }
    if (a == b && b == c) {
```

```
      return Type.EQUILATERAL;
    }
    if (a == b || b == c || a == c) {
      return Type.ISOSCELES;
    }
    return Type.SCALENE;
  }

  public static double area(int a, int b, int c) {
    if (classify(a, b, c) == Type.INVALID)
      return 0;
    double p = (a + b + c) / 2.0;
    return Math.sqrt(p * (p - a) * (p - b) * (p - c));
  }
}
```

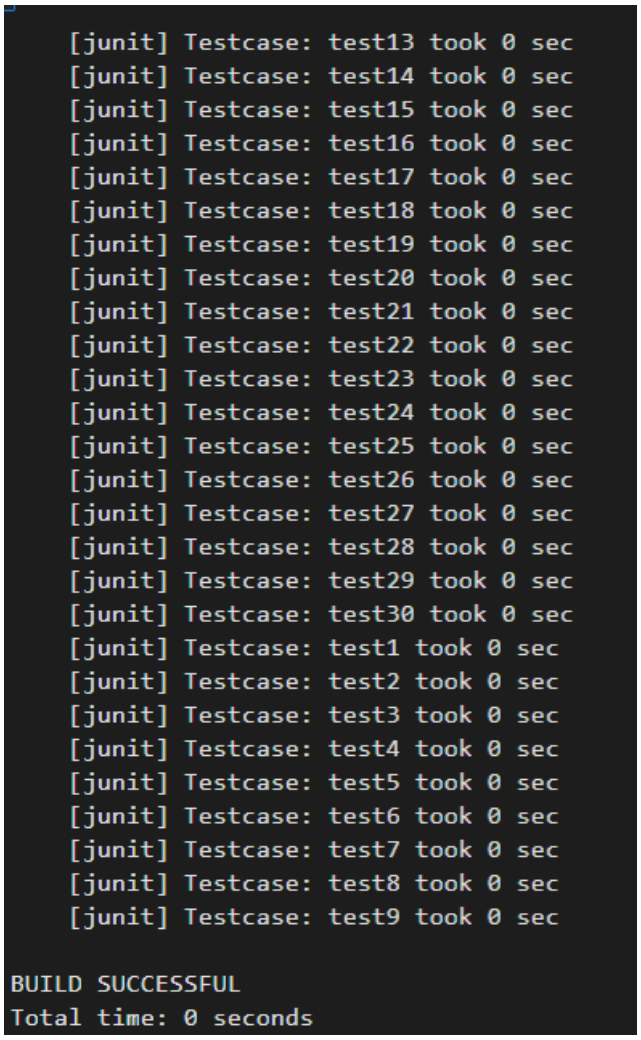
测试代码如下：（前15个测试用例测试area方法，后15个测试用例测试classify方法）

```
public class TestSuite extends TestCase {
  public void test1() {
    assertEquals(triangle.Triangle.area(0, 1301, 1), 0, 0.01);
  }

  public void test2() {
    assertEquals(triangle.Triangle.area(1108, 1, 1), 0, 0.01);
  }

  public void test3() {
    assertEquals(triangle.Triangle.area(1, 0, -665), 0, 0.01);
  }

  public void test4() {
    assertEquals(triangle.Triangle.area(1, 1, 0), 0, 0.01);
  }

  public void test5() {
    assertEquals(triangle.Triangle.area(582, 582, 582), 146671.79, 0.01);
  }

  public void test6() {
    assertEquals(triangle.Triangle.area(1, 1088, 15), 0, 0.00);
  }

  public void test7() {
    assertEquals(triangle.Triangle.area(1, 2, 450), 0, 0.00);
  }

  public void test8() {
    assertEquals(triangle.Triangle.area(1663, 1088, 823), 386411.66, 0.01);
  }
```

```
public void test9() {
  assertEquals(triangle.Triangle.area(1187, 1146, 1), 0, 0.00);
}

public void test10() {
  assertEquals(triangle.Triangle.area(1640, 1640, 1956), 1287516.10, 0.01);
}

public void test11() {
  assertEquals(triangle.Triangle.area(784, 784, 1956), 0, 0.00);
}

public void test12() {
  assertEquals(triangle.Triangle.area(1, 450, 1), 0, 0.00);
}

public void test13() {
  assertEquals(triangle.Triangle.area(1146, 1, 1146), 572.99, 0.01);
}

public void test14() {
  assertEquals(triangle.Triangle.area(1640, 1956, 1956), 1456172.24, 0.01);
}

public void test15() {
  assertEquals(triangle.Triangle.area(-1, 1, 1), 0, 0.01);
}

public void test16() {
  assertEquals(triangle.Triangle.classify(0, 1301, 1), INVALID);
}

public void test17() {
  assertEquals(triangle.Triangle.classify(1108, 1, 1), INVALID);
}

public void test18() {
  assertEquals(triangle.Triangle.classify(1, 0, -665), INVALID);
}

public void test19() {
  assertEquals(triangle.Triangle.classify(1, 1, 0), INVALID);
}

public void test20() {
  assertEquals(triangle.Triangle.classify(582, 582, 582), EQUILATERAL);
}

public void test21() {
  assertEquals(triangle.Triangle.classify(1, 1088, 15), INVALID);
}
```

```
  public void test22() {
    assertEquals(triangle.Triangle.classify(1, 2, 450), INVALID);
  }

  public void test23() {
    assertEquals(triangle.Triangle.classify(1663, 1088, 823), SCALENE);
  }

  public void test24() {
    assertEquals(triangle.Triangle.classify(1187, 1146, 1), INVALID);
  }

  public void test25() {
    assertEquals(triangle.Triangle.classify(1640, 1640, 1956), ISOSCELES);
  }

  public void test26() {
    assertEquals(triangle.Triangle.classify(784, 784, 1956), INVALID);
  }

  public void test27() {
    assertEquals(triangle.Triangle.classify(1, 450, 1), INVALID);
  }

  public void test28() {
    assertEquals(triangle.Triangle.classify(1146, 1, 1146), ISOSCELES);
  }

  public void test29() {
    assertEquals(triangle.Triangle.classify(1640, 1956, 1956), ISOSCELES);
  }

  public void test30() {
    assertEquals(triangle.Triangle.classify(-1, 1, 1), INVALID);
  }
}
```

4 结果分析



30个测试用例全部通过

```
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: Mutants generated: 115
[junit] MAJOR: Mutants covered:   115 (100.00%)
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: Export coverage map to covMap.csv
[junit] MAJOR: Export test map to testMap.csv
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: Run mutation analysis with 115 mutant(s) and 1 test(s).
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: 1/1 - triangle.test.TestSuite (3ms / 115):
[junit] MAJOR: 498 (89 / 115 / 115) -> AVG-RTPM: 4ms
[junit] MAJOR: Mutants killed (FAIL-EXC-TIME) / live: 89 (89-0-0) / 26
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: Summary:
[junit] MAJOR:
[junit] MAJOR: Analysis time                 : 0.50 seconds
[junit] MAJOR: Mutation score                : 77.39% (77.39%)
[junit] MAJOR: Mutants killed (FAIL-EXC-TIME): 89 (89-0-0)
[junit] MAJOR: Mutants live                  : 26
[junit] MAJOR: Mutant executions             : 115
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: Export summary of results to summary.csv
[junit] MAJOR: Export mutant details to details.csv
[junit] MAJOR: ---------------------------------------------------------------
[junit] MAJOR: Done!
[junit] MAJOR: ---------------------------------------------------------------
```

从上图中的变异测试结果中，不难得到：

- 所产生的突变体的数量：115
- 测试套件所覆盖的突变体的数量：115
- 被测试套件杀死的突变体的数量：89
- 活的突变体的数量：26
- 测试套件的总体突变评分/充分性：77.39%

5 问题讨论

- What do the results tell you about your test suite?
  - 测试并不能达到100%杀死突变体
- Does the test suite exhibit weaknesses? How can it be improved?
  - test suite本身有弱点，可以通过完善测试用例来提高mutation score
  - 查看mutants.log和killed.csv可以查看存活的突变体，可以有针对性的完善测试用例
- Does the test suite exhibit strengths? How do you recognize them?
  - test suite提高了测试的自动化程度，能够自动生成突变体，并对测试用例进行测试，还能够显示出存活的突变体，便于开发人员完善测试代码，提高了开发的效率，降低了测试的成本
- Do you have any other interesting insights or opinions on the experience?
  - 对于此次的实验，我遇到的最大的困难是配置环境，因为我本地的虚拟机环境被我删除，因此本次实验只好借助远程服务器来完成。在实验过程中暴露出我对Linux命令行的使用不够熟悉，今后要注重这一方面的积累。

    - 网上Major框架的相关资料较少，感觉刚开始实验时无从下手。

- Among the generated mutants, If both killed and unkilled mutants were generated, what was the type of operator used? How was it applied to the code (how did the code change)?

  如果在生成的突变体中既有被杀死的突变体，又有未被杀死的突变体，那么可能使用的是一种复杂的操作符，或者是多种不同的操作符组合生成的。这些操作符可能会对代码进行较大的改动。

  操作符的类型可能是比较常见的，如算术操作符、关系操作符、逻辑操作符等，也可能是一些较为复杂的操作符，如条件语句变异、循环语句变异等。这些操作符会对代码进行不同程度的改动，例如增加、删除、替换、重排、交换等等。

  对于被杀死的突变体，说明在测试用例下，突变后的代码行为与原代码行为不同，可能导致程序出错或无法正常执行。而未被杀死的突变体则说明在测试用例下，突变后的代码行为与原代码行为相同，因此测试用例无法检测出变异后的代码缺陷。

  因此，对于生成的突变体，需要进行全面的测试和评估，以确定其是否存在缺陷，并根据测试结果进行相应的修复和改进。

- According to your mutation analysis result, which part of the source code need to be strengthened in further coding? Which test case in your suite are more important compared with others.

  通过分析kill.csv和mutants.log。前10个变异体中有7个存活，对应代码第9行，说明此处的代码需要提高。

  ```
  if (a <= 0 || b <= 0 || c <= 0)
  ```

```
MutantNo,[FAIL | TIME | EXC | LIVE | UNCOV]
1,LIVE
2,LIVE
3,FAIL
4,LIVE
5,LIVE
6,FAIL
7,LIVE
8,LIVE
9,LIVE
10,FAIL
```

```
1:ROR:<=(int,int):<(int,int):triangle.Triangle@classify(int,int,int):9:a <= 0 |==> a
< 0
2:ROR:<=(int,int):==(int,int):triangle.Triangle@classify(int,int,int):9:a <= 0 |==>
a == 0
3:ROR:<=(int,int):TRUE(int,int):triangle.Triangle@classify(int,int,int):9:a <= 0
|==> true
4:ROR:<=(int,int):<(int,int):triangle.Triangle@classify(int,int,int):9:b <= 0 |==> b
< 0
5:ROR:<=(int,int):==(int,int):triangle.Triangle@classify(int,int,int):9:b <= 0 |==>
b == 0
6:ROR:<=(int,int):TRUE(int,int):triangle.Triangle@classify(int,int,int):9:b <= 0
|==> true
7:COR:||(boolean,boolean):!=
(boolean,boolean):triangle.Triangle@classify(int,int,int):9:a <= 0 || b <= 0 |==> a
<= 0 != b <= 0
8:COR:||
(boolean,boolean):LHS(boolean,boolean):triangle.Triangle@classify(int,int,int):9:a
<= 0 || b <= 0 |==> a <= 0
9:COR:||
(boolean,boolean):RHS(boolean,boolean):triangle.Triangle@classify(int,int,int):9:a
<= 0 || b <= 0 |==> b <= 0
10:COR:||
(boolean,boolean):TRUE(boolean,boolean):triangle.Triangle@classify(int,int,int):9:a
<= 0 || b <= 0 |==> true
```

分析99-115突变体，它们全部被杀死。能够识别出突变体并杀死的测试用例往往更加重要，对于第28行的突变体，测试用例基本上都能够杀死它们，所以，对于能够执行到28行的测试用例，也就是满足形成三角形条件，并且计算面积的测试用例，更加重要。

```
99,FAIL
100,FAIL
101,FAIL
102,FAIL
103,FAIL
104,FAIL
105,FAIL
106,FAIL
107,FAIL
108,FAIL
109,FAIL
110,FAIL
111,FAIL
112,FAIL
113,FAIL
114,FAIL
115,FAIL
```

```
99:AOR:*(double,double):/(double,double):triangle.Triangle@area(int,int,int):28:p *
(p - a) |==> p / (p - a)
```

```
100:AOR:-(double,double):%(double,double):triangle.Triangle@area(int,int,int):28:p - b |==> p % b
101:AOR:-(double,double):*(double,double):triangle.Triangle@area(int,int,int):28:p - b |==> p * b
102:AOR:-(double,double):+(double,double):triangle.Triangle@area(int,int,int):28:p - b |==> p + b
103:AOR:-(double,double):/(double,double):triangle.Triangle@area(int,int,int):28:p - b |==> p / b
104:AOR:*(double,double):%(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) |==> p * (p - a) % (p - b)
105:AOR:*(double,double):+(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) |==> p * (p - a) + (p - b)
106:AOR:*(double,double):-(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) |==> p * (p - a) - (p - b)
107:AOR:*(double,double):/(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) |==> p * (p - a) / (p - b)
108:AOR:-(double,double):%(double,double):triangle.Triangle@area(int,int,int):28:p - c |==> p % c
109:AOR:-(double,double):*(double,double):triangle.Triangle@area(int,int,int):28:p - c |==> p * c
110:AOR:-(double,double):+(double,double):triangle.Triangle@area(int,int,int):28:p - c |==> p + c
111:AOR:-(double,double):/(double,double):triangle.Triangle@area(int,int,int):28:p - c |==> p / c
112:AOR:*(double,double):%(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) * (p - c) |==> p * (p - a) * (p - b) % (p - c)
113:AOR:*(double,double):+(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) * (p - c) |==> p * (p - a) * (p - b) + (p - c)
114:AOR:*(double,double):-(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) * (p - c) |==> p * (p - a) * (p - b) - (p - c)
115:AOR:*(double,double):/(double,double):triangle.Triangle@area(int,int,int):28:p * (p - a) * (p - b) * (p - c) |==> p * (p - a) * (p - b) / (p - c)
```