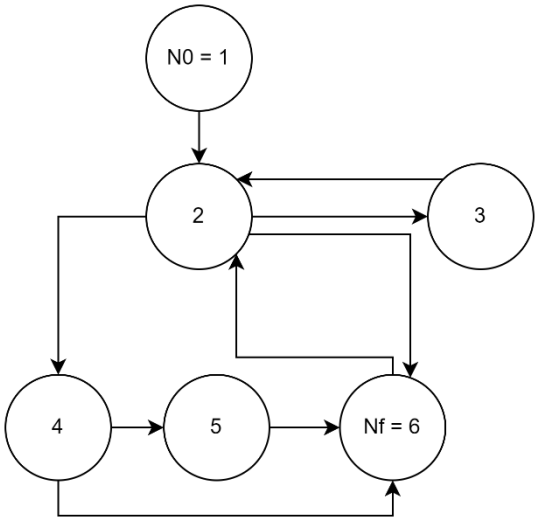
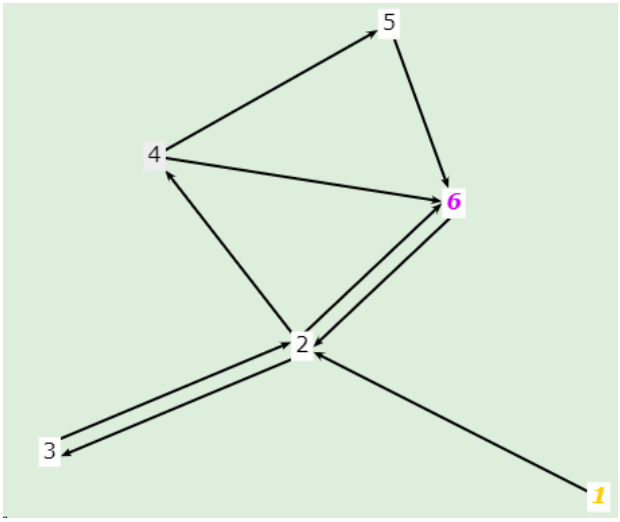
Neha Maddali

ComS417 Assignment 4

Question 1:

a. Graph (left is what I drew, right is from the textbook website):





b. There are 36 simple paths: [1,2,3], [1,2,4], [1,2,6], [2,3,2], [2,4,5], [2,4,6], [2,6,2], [3,2,3], [3,2,4], [3,2,6], [4,5,6], [4,6,2], [5,6,2], [6,2,3], [6,2,4], [6,2,6], [1,2,4,5], [1,2,4,6], [2,4,5,6], [2,4,6,2], [3,2,4,5], [3,2,4,6], [4,5,6,2], [4,6,2,3], [4,6,2,4], [5,6,2,3], [5,6,2,4], [6,2,4,5], [6,2,4,6], [1,2,4,5,6], [2,4,5,6,2], [3,2,4,5,6], [4,5,6,2,3], [4,5,6,2,4], [5,6,2,4,5], [6,2,4,5,6]

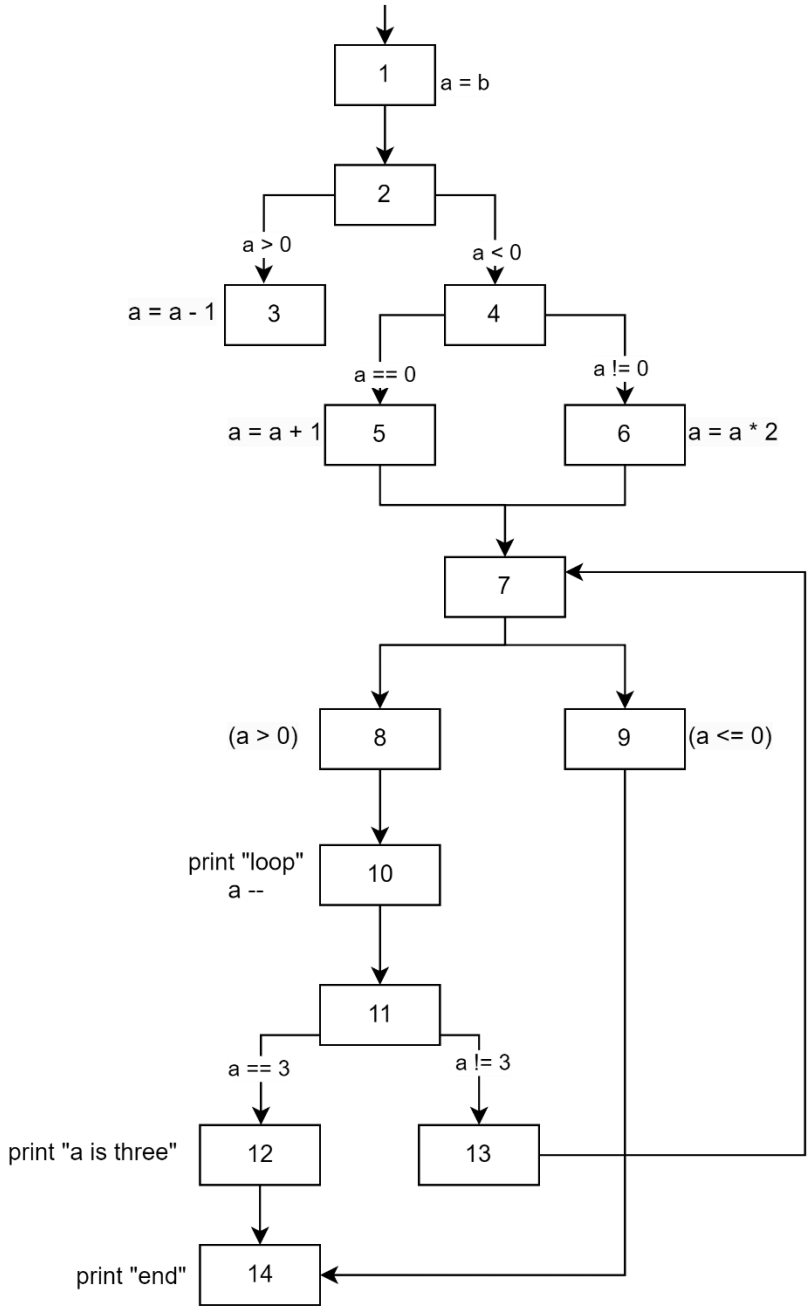
c. There are 21 prime paths: [3,2,4,5,6], [2,4,5,6,2], [1,2,4,5,6], [4,5,6,2,3], [6,2,4,5,6], [5,6,2,4,5], [4,5,6,2,4], [2,4,6,2], [3,2,4,6], [1,2,4,6], [6,2,4,6], [4,6,2,3], [4,6,2,4], [2,6,2], [3,2,3], [2,3,2], [1,2,3], [1,2,6], [6,2,6], [3,2,6], [4,4]

d. Edge coverage: there are edges [4,4], [4,6], [6,2] that are not covered
Edge pair coverage: all edge pairs are covered by the existing test paths
Node coverage: all nodes are covered by the existing test paths
To satisfy the edge coverage, add these new test paths:
   - [1,2,4,4,6]
   - [1,2,4,6,2]

Neha Maddali

Question 2:

Control flow graph:



Neha Maddali

Question 3:

1. Downloaded tar file
2. mvn verify command run
3. Number of total mutants generated: 51, killed: 29. output report:

```
================================================================================
- Timings
================================================================================
> scan classpath : < 1 second
> coverage and dependency analysis : 3 seconds
> build mutation tests : < 1 second
> run mutation analysis : 5 seconds
--------------------------------------------------------------------------------
> Total  : 9 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Generated 51 mutations Killed 29 (57%)
>> Ran 78 tests (1.53 tests per mutation)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:28 min
[INFO] Finished at: 2024-04-16T12:02:42-05:00
[INFO] ------------------------------------------------------------------------
[nmaddali@pyrite-n3 triangle-example]$
```

4. target/pit-reports/index.html:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
1	74%	17/23	57%	29/51

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
com.example	1	74%	17/23	57%	29/51

Report generated by PIT 1.4.7

5. com.example -> triangle.java lines 8-22:

```
8  6    if ((a <= 0) || (b <= 0) || (c <= 0)) {
9  1      return TriangleType.INVALID;
10     }
11     trian = 0;
12 2   if (a == b) {
13 1     trian = trian + 1;
14     }
15 2   if (a == c) {
16 1     trian = trian + 2;
17     }
18 2   if (b == c) {
19 1     trian = trian + 3;
20     }
21 2   if (trian == 0) {
22 9     if (((a + b) < c) || ((a + c) < b) || ((b + c) < a)) {
```

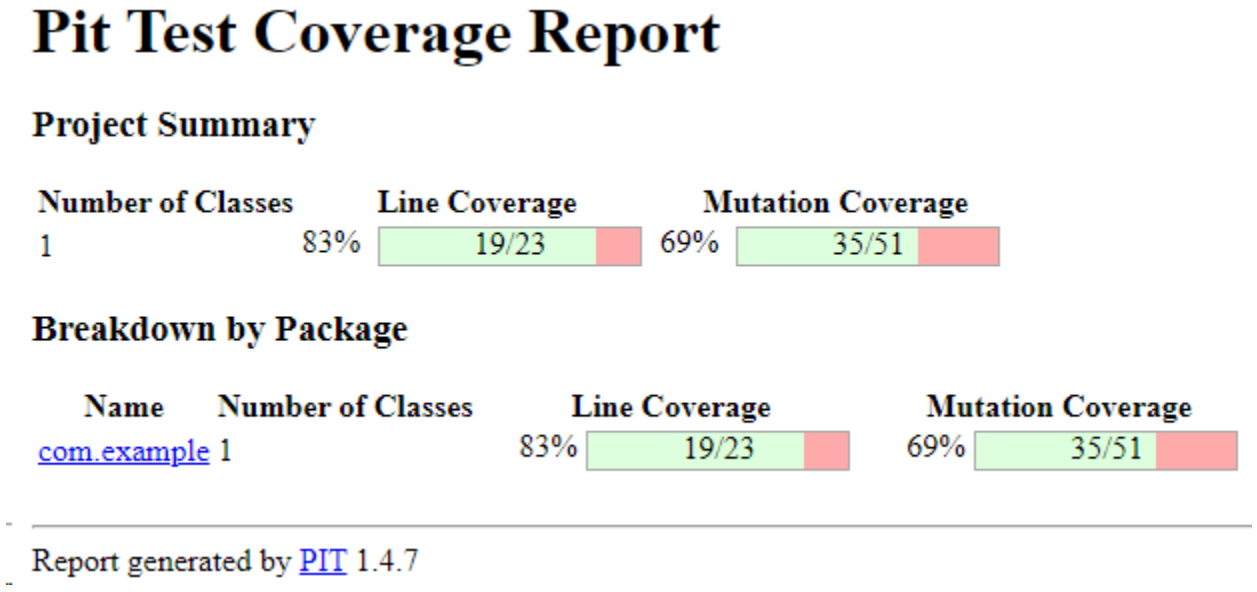
6. The “changed conditional boundary” mutation on line 8 survived. This mutation changed the conditional boundary within the if statement on line 8. It most likely altered the range of valid inputs that the program accepts, ensuring that the triangle classification function considers a broader range of inputs as valid triangles.
   This mutation survived because it maintained the logical structure of the code and didn’t introduce any errors or inconsistencies. By adjusting the conditional boundary, it potentially improved the robustness of the triangle classification function by allowing it to handle caases that were previously considered invalid.
7. I added three new test cases:

```
  @Test
  public void testIsoceles2(){
    final TriangleType type = Triangle.classify(3, 4, 3);
    assertEquals(TriangleType.ISOSCELES, type);
  }

  @Test
  public void testIsoceles3(){
    final TriangleType type = Triangle.classify(5, 3, 5);
    assertEquals(TriangleType.ISOSCELES, type);
  }

  @Test
  public void testIsocelesBoundary(){
    final TriangleType type = Triangle.classify(2147483647, 2147483647,
2147483647);
    assertEquals(TriangleType.EQUILATERAL, type);
  }
```

Here is the target/pit-report/index.html which is improved from part 4:



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
1	83%	19/23	69%	35/51

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
com.example	1	83%	19/23	69%	35/51

Report generated by PIT 1.4.7

Neha Maddali

Question 4:
Team members for the final project are Paige Rolling, Logan Ellsworth, Saiyara Iftekharuzzaman, and Jade Seiler. We are doing our final project on Large Language Models for Test Generation.