Question 1

Coverage screenshot for MoneyBagTest before modification:

lab2	94.3 %	920	56	976
test	92.7 %	507	40	547
(default package)	92.7 %	507	40	547
MoneyTest.java	0.0 %	0	23	23
MoneyBagTest.java	96.8 %	507	17	524
main	96.3 %	413	16	429

Coverage screenshot for MoneyBagTest after modification (1 method commented out):

lab2	94.3 %	862	52	914
test	93.4 %	453	32	485
(default package)	93.4 %	453	32	485
MoneyTest.java	0.0 %	0	23	23
MoneyBagTest.java	98.1 %	453	9	462
main	95.3 %	409	20	429

Coverage screenshot for MoneyTest before modification:

lab2	4.2 %	38	876	914
test	4.7 %	23	462	485
(default package)	4.7 %	23	462	485
MoneyBagTest.java	0.0 %	0	462	462
MoneyTest.java	100.0 %	23	0	23
main	3.5 %	15	414	429

This first question covers the Clover coverage software which visually shows how effective the test cases are at covering all the possible scenarios for the given main classes.

Question 2

Statement coverage Case 1 (a = 2, b = 3):
10 total statements
Line 1, 2, 3, 4, 10 covered
5/10 = 50% statement coverage

Statement coverage Case 2 (a = 3, b = 2):
10 total statements
Line 1, 5, 6, 7, 10 covered
5/10 = 50% statement coverage

Branch coverage Case 1 (a = 2, b = 3):
4 total branches
Yes/no for first if statement
yes/no for second if statement
Yes for first statement, second statement is skipped over
¼ = 25% branch coverage

Branch coverage Case 2 (a = 3, b = 2):
4 total branches
Yes/no for first if statement
yes/no for second if statement
Yes for second statement, first statement is skipped over
¼ = 25% branch coverage

Statement coverage for Q2:

lab2	6.1 %	59	914	973
test	6.9 %	36	485	521
(default package)	0.0 %	0	485	485
q2test	100.0 %	36	0	36
q2tester.java	100.0 %	36	0	36
q2tester	100.0 %	36	0	36
main	5.1 %	23	429	452
(default package)	0.0 %	0	429	429
q2	100.0 %	23	0	23
q2Function.java	100.0 %	23	0	23

Added 1 statement where a = 2 and b = 2, able to cover 100%

Question 3

Statement coverage for Q3:

lab2	13.6 %	156	992	1,148
test	14.3 %	87	521	608
(default package)	0.0 %	0	485	485
q2test	0.0 %	0	36	36
q3test	100.0 %	87	0	87
TriclassTest.java	100.0 %	87	0	87
main	12.8 %	69	471	540

System print messages:

```
Testing started
Test 4 started
Test 4 finished
Test 2 started
Test 2 finished
Test 3 started
Test 3 finished
Test 1 started
Test 1 finished
Testing is finished
```

Coverage for statements in main class:

```
        boolean rule1 = x + y > z;
        boolean rule2 = y + z > x;
        boolean rule3 = z + x > y;
        if (rule1 == false || rule2 == false || rule3 == false) {
            return "Invalid";
        }
        if (x == y && y == z) {
            return "Equilateral";
        } else if ((x == y && y != z && x != z) || (y == z && z != x && x != y) || (z == x && x != y && y != z)){
            return "Isosceles";
        } else if (x != y && y != z && x != z) {
            return "Scalene";
        } else {
```

This question showcased how coverage can help a developer visually identify what statements are being tested properly and how code can be rewritten so that there are no redundancy issues. The image provided above shows green and yellow highlights. Green highlights indicate that the statement is being covered and well used whereas the yellow highlighted indicates that the statement is only used sometimes or by certain test cases. The yellow highlights are mainly for conditional statements since not all test cases go through the conditional. Some may be rejected which means they did not pass the check. The return statements after the conditionals are green because every test case that makes it there will always run the return statement.