Problem Set 3

(10 points) Cohort Exercise 1:

Given FindMax.java, write three test cases: one resulting in Failure (does not compute maximum), one resulting in Error (throws exception) and one resulting in Pass (computes maximum).

(10 points) Cohort Exercise 2:

Given the testRepOk method in StackTest.java, decompose it into multiple **equivalent** test cases.

(10 points) Cohort Exercise 3:

Write a parameterized test for QuickSort.java.

(5 points) Cohort Exercise 6:

Write down 6 test cases for an email client (refer to the slides in Week 5). Write one test for checking successful delivery and five different tests with the intention to make the delivery fail.

(5 points) Cohort Exercise 7:

Open Disk.java. Draw the control flow graph of the function manipulate().

(10 points) Cohort Exercise 8:

Open Disk.java. Write a set of tests to cover each statement of the manipulate() function. How many tests did you write? Is it the minimum number of tests to cover all the statements?

(10 points) Cohort Exercise 9:

Open Disk.java. Write a set of tests to cover each branch of the manipulate() function. How many tests did you write? Is it the minimum number of tests to cover all the branches?

(10 points) Cohort Exercise 10:

Open Disk.java. Assume that the loop in the manipulate() function executes at most 100 times. How many paths are there in the manipulate() function? Explain your answer.

(5 points) Cohort Exercise 11:

Open Disk.java. Consider your test cases that obtain branch coverage in the manipulate() function. Argue whether the test suite also obtains the condition coverage.

(10 points) Cohort Exercise 12:

Consider Disk.java. Assume that specification requires all the functions in the Disk class to be terminating. Write a Junit test that potentially reveals a bug in the manipulate() function.

(15 points) Cohort Exercise 13:

Given method multiply Russian.java,

- Create a test suite for black-box testing.
- Create a test suite for white-box testing
  - With 100% branch coverage
- Create a test suite for fault-based testing