Problem Set 2

(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).

(5 points) Cohort Exercise 2:

Fix the class Stack (in Stack.java) so that all tests in StackTest.java pass.

(5 points) Cohort Exercise 3:

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

(10 points) Cohort Exercise 4:

Write a parameterized test for QuickSort.java.

(5 points) Cohort Exercise 5:

Write down six Blackbox test cases for an email client. Write one test for checking successful delivery and five different tests with the intention to make the email delivery fail. You do not need to write JUnit tests for this. Think the email client as a blackbox and determine the set of input fields (e.g. email, SMTP server name etc.) that the client can take. A test will capture values in the different input fields for the client.

(5 points) Cohort Exercise 6:

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

(10 points) Cohort Exercise 7:

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

(10 points) Cohort Exercise 8:

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

(10 points) Cohort Exercise 9:

Assume that the loop in the manipulate() function is terminated after **at most** 100 iterations (i.e. after 0 iteration, 1 iteration, …., 100 iterations etc.). Based on this assumption, compute the possible number of executed paths in the manipulate() function. Explain your answer.

(5 points) Cohort Exercise 10:

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 11:

Consider Disk.java. Assume that specification requires all the functions in the Disk class to be terminating. Write a test (i.e. values of x and y) that potentially reveals a bug in the manipulate() function.

(15 points) Cohort Exercise 12:

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

For all the above test cases for Russian.java, write JUnit tests.