Problem Set 3

(10 points) Cohort Exercise 1:

Given FindMax.java, write three test cases: one resulting in Failure, one resulting in Error and one resulting in Pass.

(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.

(10 points) Cohort Exercise 4:

Given Sorter.java and FindMaxUsingSorting.java, write a test case for method FindMax without implementing Sorter.java first.

(10 points) Cohort Exercise 5:

Work in pairs. One writes a set of black-box tests for the program implementing the following specification: *Given an integer value, the program outputs 0 when the integer value is 0, outputs 1 when the integer value > 0, and outputs -1 when the integer value < 0.*

The other implements the programs with the intention to pass the test cases while not satisfying the specification.

(10 points) Cohort Exercise 6

Work as a group, write down 6 test cases for your game.

(10 points) Cohort Exercise 7

Given BiSectionExample.java, create a test suite which covers “every statement”.

(10 point) Cohort Exercise 8

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

(20 points) Homework Question 1:

(Work as a team) Program the game logic class for your game. The class is supposed to maintain the game status and to support various operations which update the game status. Document the specification of the class (informally using program comments) and each of the methods in the class. Program a set of unit test cases for each method to achieve 100% branch coverage.