Question 1

This question tackles combining 2 arrays together by multiplying the values of the shorter array with the corresponding values in the longer array then filling the rest of the array positions with the longer array values. The code first figures out the length of the shorter array and identifies the longer array. Then using 2 for loops, iterates through the length of the short one, multiplying the corresponding values from each array, and then filling the return array with the rest of the values from the longer array. JUnit test case just assertEquals the array to ensure the correctness of the values coming out of the written method mult().

Question 2

This question had an error already with the given code. Heron's equation was then found on the internet and fixed in the code so that there were no more compilation errors. The JUnit test class includes the following methods listed in order, testing validity of Triangle(3, 4, 100), testing the method calculateArea with values that were known, testing if t1 and t2 are equal, and a test showing what happens if length is negative and showing that the code cannot be misused.

Question 3

This question tackles regex, using a combination of symbols in a string to help the compiler understand what characters you are trying to isolate and identify for. These symbols are used as delimiters to segment off what characters are trying to be identified in that specific sequence. With the given regex:

^\\s*[(]*(\\d{3})[)][ ]*(\\d{3})[-]*[-]*[ ]*(\\d{4})$

- ^\\s shows the start of a line
- * shows that the symbols used before it are optional meaning they are not a requirement for the string to return true
- [(] show the symbols inside the square brackets are the ones that are used to identify
- \\d{3} show that we are trying to identify 3 digits
- [)] show the symbols inside the square brackets are the ones that are used to identify, there is also a space inside the brackets that indicate we are trying to identify a space
- $ dictates the end of the line

The test cases in this question simply just test whether or not the formats of phone numbers provided in the lab manual test valid.

Question 4

This question helps teach about JUnit Suites which are just an empty class that house the JUnit test classes. This allows for tidiness in coding and provides an organised location where all the tests for the specified project can be run with a click of a button.

Question 5

This question tackles parameterized testing with a constructor which simply means the testing class that has a test method and another method that contains the preset parameters that will be used to test with. There is a constructor in this test class that allows the parameters to be fed into this test class each time there is a new value to test for aka the next value in the test array. The parameters provided is a 2D array which each element being an array with 2 elements, the number which fibonacci sequence we are looking for, and then known number at the nth fibonacci sequence.

Question 6

This question follows the same format as Question 5 but with testing for prime numbers. The parameters in this question are the numbers that are used to be tested along with their results whether it's true or false.

Question 7

This question introduces Junit Theories which is similar to parameterized testing except there are more limitations. Theories are similar to theories in Math which provide a statement that is true to certain values and are tested against other values to prove its validity. In the beginning of the question, a theory is provided that a + b > a and a + b > b. This theory was then set in the code and a set of values were provided to test the validity of the theory. The theory contains an assertTrue test which ensures that the statement given inside the assertion is true otherwise it will halt with a fail.

Question 8

This question just converts the parameterized testing in question 5 and 6 into theory JUnit tests. Instead of setting parameters for a test, a theory is made so that the result of the method is equal to the value given in the set of values. A set of 2 value arrays

were provided with the first being the test value and the second being the asserted value. This is very similar to the code in the parameterized testing but instead uses theories.