Q1. Suppose you have the following class which you want to test:

```
public class Calculation {
        public static int findMax(int arr[]){
                int max=arr[0];
                for(int i=1;i<arr.length;i++){
                        if(max<arr[i])
                                max=arr[i];
                }
                return max;
        }
}
```

findMax is a function that accepts an array of integers as its input parameter. It returns the largest value among the values in the input array. Write a Junit unit test for the findMax function. The requirement is: Use an array with values 1, 3, 4, 2 as the test case.

Answer:

```
public class TestFindMax {

    @Test    (1 marks)

    public void testFindMax(){
        assertEquals(4,Calculation.findMax(new int[]{1,3,4,2}));    (assertEquals , 2 marks, the function
parameters 2 mark)
    }
}
```

Q2. Suppose you have the following Calculator class which you want to test.

```
public class Calculator {
        public int multiply(int a, int b) {
                return a * b;
        }
}
```

Fill up the following test class to complete the testing for the above Calculator class. Additional requirements are:

- Use 4* 5 = 20 as the test case
- The test should run 5 times
- Make use of all imported classes.

```
import static org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.RepeatedTest;
import org.junit.jupiter.api.Test;

class CalculatorTest {
...
        void setUp() {
                ...
        }
      void testMultiply() {
           ...
        }
      }
}
```

Answer:

```
import static org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.RepeatedTest;
import org.junit.jupiter.api.Test;

class CalculatorTest {

    Calculator calculator; (1 mark)

    @BeforeEach (1 mark)
    void setUp() {
        calculator = new Calculator(); (1 mark)
    }

  @Test (1 mark)
  @RepeatedTest(5)          (1 mark)
  @DisplayName("Simple multiplication should work")    (1 mark, any output is accepted)
    void testMultiply() {
        Assertions.assertEquals(20, calculator.multiply(4, 5),
                "Regular multiplication should work");
(4 marks, 2 for Assertions.assertEquals , 2 for the function parameters )
    }
}
```