Week 2 – Test Case Instructions

TODO: Determine where to put warmup Java files.

For the Chapter 2 autograded programming projects, the Revel autograder expects a class named **Exercise**. If you create a class with a different name in an IDE, remember to change the name to **Exercise** when submitting to the Revel autograder.

Instructions:

- Download and edit this document to fill in your test plans.
- Follow the Course Resource lesson "Week 2 Specification-based Testing" to create test plans:
    - Warmup activity BuggyWarmup1.java.
    - Warmup activity BuggyWarmup2.java.
    - Chapter 2 Programming Project 1.
    - Chapter 2 Programming Project 5.
- When editing test cases, fill in the expected I/O (input/output) **prior** to running the program. Fill in the actual I/O and status (Pass or Fail) after running the program.
    - Display the input values in bold. It's ok to wrap a line of output across multiple lines if the table cell is not wide enough to display it on one line. Assume input is entered on the same line as the prompt, even if it is displayed on a separate line in the test case.
    - Refer to the Revel sample runs for exact input/output expectations per project.
- Describe the lessons learned while implementing each assigned programming project. What challenges did you encounter and how did you solve them?

Note: This submission is only for the test plans. The Java code for your programming project solutions should be submitted in the Revel environment for grading.

Warmup Activity #1 – BuggyWarmup1.java

```java
import java.util.Scanner;

/**
 * BuggyWarmup1 reads two numbers and prints the average.
 */
public class BuggyWarmup1 {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter 2 numbers: ");
        int num1 = input.nextInt();
        int num2 = input.nextInt();
        double avg = num1 + num2 / 2.0;
        System.out.println("Average = " + avg);
    }
}
```

Run BuggyWarmup1.java for each test case listed below. Fill in the actual output and status (Pass/Fail).

	Expected I/O	Actual I/O	Status
1	Enter 2 numbers: **10 24** Average = 17.0		
2	Enter 2 numbers: **13 42** Average = 27.5		

There is a calculation error in BuggyWarmup1.java. Identify and fix the error so the actual and expected output match. Rerun the tests to confirm your solution is correct.

	Expected I/O	Actual I/O	Status
1	Enter 2 numbers: **10 24** Average = 17.0		
2	Enter 2 numbers: **13 42** Average = 27.5		

Include a screen print of your code solution:

Warmup Activity #2 – BuggyWarmup2.java

R0 (R-naught) is a term to describe the reproduction rate of infectious pathogens.

If a disease has an R0 of 2, each infected person spreads the disease to approximately 2 others. After 3 iterations, we would expect 2^3= 8 infections. In comparison, an R0 of 5 results in 5^3= 125 infections.

The program BuggyWarmup2.java reads in a value for R0 and attempts to compute the number of infections after 3 iterations, i.e. $\mathbf{R0}^3$. However, the code contains an error. Refer to the documentation for **Math.pow** at https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#pow-double-double-

Run BuggyWarmup2.java for each test case listed below. Fill in the actual output and status (Pass/Fail).

	Expected I/O	Actual I/O	Status	Disease
1	Enter R0: **2** 3 iterations result in 8 infections			Ebola R0 = 2
2	Enter R0: **3** 3 iterations result in 27 infections			Sars R0 = 3
3	Enter R0: **5** 3 iterations result in 125 infections			Polio R0 = 5

Identify and fix the error in BuggyWarmup2.java so the actual and expected output match.
Rerun the tests to confirm your solution is correct.

	Expected I/O	Actual I/O	Status	Disease
1	Enter R0: **2** 3 iterations result in 8 infections			Ebola R0 = 2
2	Enter R0: **3** 3 iterations result in 27 infections			Sars R0 = 3
3	Enter R0: **5** 3 iterations result in 125 infections			Polio R0 = 5

Include a screen print of your code solution:

Chapter 2 Project 1 Test Plan

Fill in the expected I/O for test cases 3 and 4 based on the values specified in the comment. Define two additional test cases 5 and 6, describing your choice of input values in the comment. Run your Project 1 code solution for each test case and fill in the actual I/O and status based on the program execution. Fix any errors in your code.

	Expected I/O	Actual I/O	Status	Comment
1	Enter the subtotal and a gratuity rate: **10 15** The gratuity is $1.5 and total is $11.5			Sample run
2	Enter the subtotal and a gratuity rate: **12.8 10.5** The gratuity is $1.344 and total is $14.144			subtotal $12.8 rate 10.5%
3				subtotal $0 rate 20%
4				subtotal $27.50 rate 0%
5				
6				

Lessons learned while implementing programming project 1:

Chapter 2 Project 5 Test Plan

You should be able to predict the expected I/O for test case #3 based on the values from tests #1 and #2.

Come up with input values for test cases 5 – 7, using a positive number for any variable not assigned to 0 in the comment. Fill in the expected I/O for test cases 5-7 prior to running the program.

Run your Project 5 code solution for each test case and fill in the actual I/O and status based on the program execution. Fix any errors in your code.

	Expected I/O	Actual I/O	Status	Comment
1	Enter investment amount: **1000.56** Enter annual interest rate in percentage: **4.25** Enter number of years: **1** Accumulated value is $1043.92			Sample run. $1000.56 investment, 4.25%, 1 year
2	Enter investment amount: **1043.92** Enter annual interest rate in percentage: **4.25** Enter number of years: **1** Accumulated value is $1089.16			$1043.92 investment, 4.25%, 1 year
3				$1000.56 investment, 4.25%, 2 years
4	Enter investment amount: **2000** Enter annual interest rate in percentage: **5** Enter number of years: **10** Accumulated value is $3294.01			$2000 investment, 5%, 10 years
5				0 years
6				0 rate
7				0 investment

Lessons learned while implementing programming project 5: