CS265FZ Software Testing
Lab 3 – Combinational Testing

There are TWO exercises to be completed.

Two pieces of work need to be submitted:

1. Fill in this lab sheet and submit it to Moodle. You don't need to attach your source code in this form. You need to upload your source code separately.
2. Submit all the required source code to Moodle. Make sure your source code is tested in Eclipse and is executable.
3. Make sure you provide detailed comments in the source code:
    a. Identify the fault(s) in the source code
    b. How did you fixed the fault(s)?

Problem 1

A program (Source Code: ***Lab3_Program1.java***) is used by an airline company to automatically assess the level of insurance the customer must pay on their ticket. Each customer can bring one piece of sports equipment and one piece of musical equipment on a flight:

If they bring both sports and music equipment, the insurance is **€20**

If they only bring one piece of equipment, the insurance is **€10**

If they bring no equipment, then the insurance fee is **€5**

The program input consists of two **boolean** variables:

1. ***sportsEquipment***
2. ***musicEquipment***

The program output is a single variable: ***insurance***

Task 1

Identify the *causes* and *effects*. Based on the *causes* and *effects* identified, draw a truth table for the combinational testing (In order to reduce the size of the truth table, ignore invalid inputs). From the truth table, generate test data for the test. (*NOTE: show the process for reducing the number of causes.)

Causes

sportsEquipment and ***musicEquipment*** are true
sportsEquipment or ***musicEquipment*** is true
sportsEquipment and ***musicEquipment*** are false

Effects

Return the ***insurance*** value 20
Return the ***insurance*** value 10
Return the ***insurance*** value 5

Truth Table

Rule 1: If ***sportsEquipment*** and ***musicEquipment*** are true, then return the ***insurance*** value 20.
Rule 2: If ***sportsEquipment*** or ***musicEquipment*** is true, then return the ***insurance*** value 20.
Rule 3: If ***sportsEquipment*** and ***musicEquipment*** are false, then return the ***insurance*** value 5.

		Rule			
		1	**2**		**3**
cause	sportsEquipment	T	T	F	F
	musicEquipment	T	F	T	F
effect	20	T	F	F	F
	10	F	T	T	F
	5	F	F	F	T

Each Rule is a Test Case.

Test Data

Test ID	**Test Cases/ Rules Covered**	**Inputs**		**Expected Outputs**
		sportsEquipment	***musicEquipment***	***Insurance***
T1.1	CT-1	true	true	20
T1.2	CT-2	true	false	10
T1.3	CT-[2]	false	true	10
T1.4	CT-3	fasle	false	5

Task 2

Based on the specification given above, write your testing code in JUnit 5 to test the source code of the program provided on Moodle ("***Lab3_Program1.java***"). Make sure your test code is named as "***Lab3_Task1.java***".

Task 3

Based on the test results, provide the correct version of the "***Lab3_Program1.java***", and rename it to "***Lab3_Program1_Fix.java***".

Problem 2

A store in a city offers different discounts depending on the purchases made by the individual (Source Code: ***Lab3_Program2.java***). To test the software that calculates the discounts, it is possible to identify the ranges of purchase values that earn the different discounts. For example, a purchase in the range of (€0 to €50] has no discount, a purchase over €50 to €200 has a 5% discount(€50 to €200], and purchases of over €200 inclusive to €500 have a 10% discounts(€200 to €500], and purchases of over €500 have a 15% discount. Invalid input will have a return value of 0 as discount.

Note: discounts are presented as follows: no discount is presented with 0, discount of 5% with 0.05, discount of 10% with 0.1, and 15% discount is presented as 0.15.

Task 1

Identify the *causes* and *effects*. Based on the *causes* and *effects* identified, draw a truth table for the combinational testing (In order to reduce the size of the truth table, ignore invalid inputs). From the truth table, generate test data for the test. (*NOTE: show the process for reducing the number of causes.)

Causes

Invalid **Purchase** values smaller than 0
Purchase values in the range of (0, 50]
Purchase values in the range of (50, 200]
Purchase values in the range of (200, 500]
Purchase values over 500

Effects

Return the **discount** value **0**
Return the **discount** value **0.05**
Return the **discount** value **0.1**
Return the **discount** value **0.15**

Truth Table

Rule 1: If ***Purchase*** values are smaller than 0**,** then return the ***discount*** value **0**
Rule 2: If ***Purchase*** values are in the range of (0, 50]**,** then return the ***discount*** value **0**
Rule 3: If ***Purchase*** values are in the range of (50, 200]**,** then return the ***discount*** value **0.05**
Rule 4: If ***Purchase*** values are in the range of (200, 500]**,** then return the ***discount*** value **0.1**
Rule 5: If ***Purchase*** values are over 500**,** then return the ***discount*** value **0.15**

		Rule				
		1	**2**	**3**	**4**	**5**
cause	Purchase<=0	T	F	F	F	F
	0<Purchase<=50	F	T	F	F	F
	50<Purchase<=200	F	F	T	F	F
	200<Purchase<=500	F	F	F	T	F
	Purchase>=500	F	F	F	F	T
effect	0	T	T	F	F	F
	0.05	F	F	T	F	F
	0.1	F	F	F	T	F
	0.15	F	F	F	F	T

Each valid Rule is a Test Case.

Test Data

Test ID	**Test Cases/ Rules Covered**	**Inputs**	**Expected Outputs**
		purchase	*discount*
T2.1	CT-1	0	0
T2.2	CT-2	1	0
T2.3	CT-3	51	0.05
T2.4	CT-4	201	0.1
T2.5	CT-5	501	0.15

Task 2

Based on the specification given above, write your testing code in JUnit 5 to test the source code of the program provided on Moodle ("***Lab3_Program2.java***"). Make sure your test code is named as "***Lab3_Task2.java***".

NOTE: when using the ***assertEquals( )*** function to compare two floating-point numbers, you need to use the following format:
assertEquals(ExpectedResults, ProducedResults, EPSILON); where EPSION represents a very small number, for example, ***1.0E-6***.

Task 3

Based on the test results, provide the correct version of the "***Lab3_Program2.java***", and rename it to "***Lab3_Program2_Fix.java***".