# CSC364/CSCM64 Lab 2 with Answer Suggestions

**Task 1.**
Consider the following specification of a computational problem:

**Altered Product.**

- **Input:** Two integers $x, y$.
- **Output:**
  - `Out of Range` if any of the inequalities $1 \leq x \leq 10$ or $2 \leq y \leq 12$ is violated.
  - $x \cdot y - 2$ otherwise.

**Tasks.**

1. Give a test suite for Boundary Value Analysis for the problem Altered Product.
2. Briefly describe how your test suite would change when applying Robustness Testing, Worst Case Testing, or Robust Worst Case Testing. How many test cases do you get with each respective testing method?
3. Which of the four variations of Boundary Value Testing do you think is most suitable for this problem? Give a brief justification of your answer.

**Answer 1.** 1. We choose 5 as a "nominal" value for $x$ and 6 as a "nominal" value for $y$. A Boundary Value Analysis test suite is then given as follows:

| Case | a | b | Expected Output |
|---|---|---|---|
| 1 | 1 | 6 | 4 |
| 2 | 2 | 6 | 10 |
| 3 | 9 | 6 | 52 |
| 4 | 10 | 6 | 58 |
| 5 | 5 | 1 | 3 |
| 6 | 5 | 2 | 8 |
| 7 | 5 | 11 | 53 |
| 8 | 5 | 12 | 58 |
| 9 | 5 | 6 | 28 |

2. When applying Robustness Testing we would add four more test cases: two where $x$ ranges over 0 and 11 respectively while $y$ is kept at its “nominal” value of 6, and two where $y$ ranges over 0 and 13 respectively, while $x$ is kept at its “nominal”value of 5. The expected outputs would be `Out of Range` for all new test cases.

   For Worst Case Testing we would have $x$ and $y$ range over all possible combinations of boundary, near-boundary, and nominal values, yielding $5^2 = 25$ test cases in total. For Robust Worst Case Testing we would include the near-boundary values outside the domain, yielding $7^2 = 49$ test cases in total.

3. The test suite for Robust Worst Case Testing yields the best coverage since it contains all other test suites. Since its size is not too large, one can justify using this strategy.

   On the other hand, the Independent Fault Assumption is likely satisfied here, so Robustness Testing will likely find the same faults as Robust Worst Case Testing. The size of the test suite for Robustness Testing is less than a third of that for Robust Worst Case Testing, so the amount of effort saved is significant.

   There do not seem to be any serious drawbacks in using Robustness Testing over Boundary Value Analysis.

   Thus, one can justify both Robust Worst Case Testing and Robustness Testing. Robustness Testing is probably slightly preferable, since it requires less effort.

**Task 2.**

Consider the following specification of a computational problem:

**Line Configuration Classification.**

- **Input:** Six integers $a, b, c, u, v, w$
- **Output:**
  - `Out of Range`, if at least one of the six inputs lies outside the interval $[0, 100]$.
  - `Invalid Input`, if $a = b = 0$ or $u = v = 0$.
  - `Equal`, if the input is neither "out of range" nor "invalid" and if the straight line $L_1$ defined by the equation $ax + by = c$ is equal to the straight line $L_2$ defined by the equation $ux + vy = d$.
  - `Parallel`, if the input is neither "out of range" nor "invalid" and if the straight line $L_1$ defined by the equation $ax + by = c$ is parallel but not equal to the straight line $L_2$ defined by the equation $ux + vy = d$.
  - `Intersecting` in all other cases.

**Tasks**

1. Give a test suite for Boundary Value Analysis for the problem Line Configuration Classification. You are allowed to use symmetry properties of the problem to reduce the number of test cases.
2. Discuss the quality of your test suite. Is it likely to detect common faults that are present in a software solution to the problem? Is it likely to detect most faults? Does it provide good coverage?
3. Briefly discuss the benefits and drawbacks of Robustness Testing, Worst Case Testing, and Robust Worst Case Testing for this problem. Do they yield better test suites than Boundary Value Analysis? Which of the four variants of Boundary Value Testing is best suited for generating a test suite for this problem?

**Answer 2.** 1. We pick the nominal value 50 for all variables. The values that our variables will range over are hence:

$$0, 1, 50, 99, 100.$$

Half a test suite (13 cases) is given below. The other half (12 more cases) is symmetrical. One can further reduce the size of the test suite by 4 cases to a total of 8 cases by observing that the roles of $a$ and $b$ are symmetrical.

**NB.** We are applying these "size reductions" to save ourselves some work when writing down the test suite. When testing a system we would normally apply the full test suite, since we cannot rely on the software implementing the symmetries of the problem correctly. On the other hand, if we do know for some reason that the software does take certain symmetries into account correctly, we can actually run a smaller test suite to save costs.

| Case | a | b | c | u | v | w | Expected Output |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 50 | 50 | 50 | 50 | 50 | `Intersecting` |
| 2 | 1 | 50 | 50 | 50 | 50 | 50 | `Intersecting` |
| 3 | 99 | 50 | 50 | 50 | 50 | 50 | `Intersecting` |
| 4 | 100 | 50 | 50 | 50 | 50 | 50 | `Intersecting` |
| 5 | 50 | 0 | 50 | 50 | 50 | 50 | `Intersecting` |
| 6 | 50 | 1 | 50 | 50 | 50 | 50 | `Intersecting` |
| 7 | 50 | 99 | 50 | 50 | 50 | 50 | `Intersecting` |
| 8 | 50 | 100 | 50 | 50 | 50 | 50 | `Intersecting` |
| 9 | 50 | 50 | 0 | 50 | 50 | 50 | `Parallel` |
| 10 | 50 | 50 | 1 | 50 | 50 | 50 | `Parallel` |
| 11 | 50 | 50 | 99 | 50 | 50 | 50 | `Parallel` |
| 12 | 50 | 50 | 100 | 50 | 50 | 50 | `Parallel` |
| 13 | 50 | 50 | 50 | 50 | 50 | 50 | `Equal` |

2. The test suite does have the potential of detecting certain common faults:

   - It is possible that the code does not check the upper and lower bounds correctly. This is likely detected by the test suite.
   - It is possible that the code involves computation of the slope of one of the lines or both the lines by computing quantities such as $a/b$, $b/a$, $u/v$, or $v/u$. There is a potential for division by zero, which would be detected by the test suite in all four cases.

   On the other hand, many interesting configurations of lines that may be handled incorrectly are left out. For example:

   - All types of configuration where both lines are parallel to one of the coordinate axes.
   - Configurations where lines are parallel but close to equal.
   - Configurations where lines are equal but given by different equations.
   - Configurations where lines are parallel but the equations have different left-hand sides.

- Configurations where one or both equations do not specify a straight line (observe that the output `Invalid Input` is missing from our test suite).

Overall, the test suite could serve as a reasonable complement to a more problem-specific test suite. The coverage is not great, but there are certain faults that it will reliably catch. It is fairly easy to generate and to execute. On its own, the test suite would not provide sufficient coverage to demonstrate likely absence of faults.

3. Robustness Testing would add 12 more test cases, increasing the total number of test cases from 25 to 37. The expected outcome for all added test cases is `Out of Range`. The number of added test cases is fairly small, they are easy to create, and they strictly improve the coverage of our test suite. Therefore it is probably a good idea to add them.

   A test suite for Worst Case Testing will have $5^6 = 15,625$ test cases. This makes manual generation of the test suite infeasible. A Worst Case Testing test suite would cover some of the cases the original test suite missed, for instance the invalid configurations or the configurations where both lines are parallel to the coordinate axes. This however comes at the cost of a very large number of mostly redundant test cases. A similar remark applies to Robust Worst Case Testing – a test suite would have $7^6 = 117,649$ test cases.

   In conclusion, Robustness Testing seems to be the most appropriate choice, since it yields slightly better coverage than Boundary Value Analysis and is not much more expensive to create or to run.

**Task 3.**
Consider the following specification of a computational problem:

**Mortgage.**

- **Input:**
  - An integer `age`.
  - An integer `salary`.
  - A enumerated value `gender` $\in$ {`male`, `female`}.
- **Output:**
  - `Out of Range`, if `age` does not fall within the closed interval [18,55] or `salary` does not fall within the closed interval [1,10000].

– Otherwise, an integer `salary` · `factor`, where `factor` is computed according to the following table:

| **Category** | `factor` |
|---|---|
| Young (18 – 35 years) | 75 for `male`, 70 for `female` |
| Middle (36 – 45 years) | 55 for `male`, 50 for `female` |
| Old (46 – 55 years) | 30 for `male`, 35 for `female` |

**Tasks**

1. Define partitions of the input domain for the Mortgage problem.
2. Give a minimal test suite for Weak Normal Equivalence Class testing.
3. Give a minimal test suite for Strong Normal Equivalence Class testing (you are allowed to reuse values from your previous test suite).
4. Which of these two testing methods is better suited for the problem? Provide a short justification for your answer.

**Answer 3.** 1. Partition `age` into the three intervals $Y = [18, 35]$, $M = [36, 45]$, and $O = [46, 55]$. The variable `gender` is partitioned into two sets, {`male`} and {`female`}. The variable `salary` can be partitioned into a single equivalence class [1, 10000].

The classes were chosen this way because different combinations of age and gender lead to different behaviours of the system. On the other hand, the system arguably behaves the same for all values of the salary.

2.

| Case | `age` | `salary` | `gender` | Expected Output |
|---|---|---|---|---|
| 1 | 21 | 20 | `male` | 1500 |
| 2 | 40 | 20 | `male` | 1100 |
| 3 | 46 | 20 | `female` | 700 |

3.

| Case | `age` | `salary` | `gender` | Expected Output |
|---|---|---|---|---|
| 1 | 21 | 20 | `male` | 1500 |
| 2 | 40 | 20 | `male` | 1100 |
| 3 | 46 | 20 | `female` | 700 |
| 4 | 21 | 20 | `female` | 1400 |
| 5 | 40 | 20 | `female` | 1000 |
| 6 | 46 | 20 | `male` | 600 |

4. Strong Normal Equivalence Testing is more appropriate here. The test suite has only a small number of test cases, and it yields better coverage, as it tests for all possible combinations of age and gender. Observe that all different combinations of age and gender lead to different behaviours of the system. By contrast, Weak Normal Equivalence Testing only tests half of all combinations.