# Homework: Test Techniques I

## Equivalence Partitioning / Boundary Value Analysis – Income Checker

Now that you are familiar with the Equivalence Partitioning / Boundary Value Analysis Techniques, let's recall [**The Income Checker App**](http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/income-checker/) from the QA Basics course, that categorizes the given **monthly income** into one of the following categories: "**low**", "**mid**", "**high"**. It works as follows:

* If the income is less than 1000, returns “**low**”
* If the income between 1000 (inclusively) and 3000 (exclusively), returns “**mid**”
* If the income is equal or bigger than 3000, returns “**high**”
* If the income is negative, returns “**error**”

**Your task is:**

**Equivalence Partitioning:** Divide the possible input values of the "**income**" into different equivalence classes or partitions. Remember to include both valid and invalid partitions.

**Boundary Value Analysis:** Identify the boundary values of the defined partitions and come up with test cases that include these boundary values. Ensure you consider "**edge cases**" - values just outside of valid ranges.

**Note:** Keep in mind that testing should cover not only expected or valid inputs but also unexpected or invalid ones. Consider all possible scenarios that might be encountered in a real-world situation.

**Equivalence Partitioning Test Cases including invalid cases:**

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Input** | **Expected Output** |
| TC01 | -1 | “error” |
| TC02 | 0 | "low" |
| TC03 | 999 | “low” |
| TC04 | 1000 | "mid" |
| TC05 | 2999 | "mid" |
| TC06 | 3000 | “high” |

**Boundary Value Analysis Test Cases including invalid cases:**

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Input** | **Expected Output** |
| TC07 | 999 | "low" |
| TC08 | 1000 | "mid" |
| TC09 | 1001 | “mid” |
| TC10 | 0 | “low” |
| TC11 | 1 | “low” |
| TC12 | -1 | “error” |
| TC13 | 3000 | “high” |
| TC14 | 2999 | “mid” |
| TC15 | 3001 | “high” |

## 2. Pairwise Testing - eCommerce Checkout Function

Assume you have a checkout function of an eCommerce application for testing. The function contains the following fields with their input values:

**Drop-down menu that contains 5 different shipping methods (input values – 1, 2, 3, 4, 5);**

**Radio button for gift wrapping (input values – Yes or No);**

**Checkbox for agreeing to terms and conditions (input values - Checked or Unchecked);**

**Place Order button (input values - Does not accept any value, only finalizes the order).**

**Your task is:**

1. Calculate how many would be the positive test cases, if you have to cover every single possibility?

|  |
| --- |
| **Your Answer: 5(shipping methods) \* 2(gift wrapping options) \* 2(checkbox options) = 20** |

Using Pairwise testing, reduce the number of necessary test cases.

|  |
| --- |
| **Add a screenshot of the reduced test cases here:** |

We have only considered positive test cases so far. What about negative ones? Write at least 3 negative test cases.

|  |
| --- |
| Example: Attempt to place an order with no shipping method selected. |
| **TC01**: An input value is 9 for the shipping method. This is not a valid option. An error message should display. |
| **TC02**: If we enter a value “Hello” for the gift wrapping. The system would display an error message or to prompt user to select “yes” or “no” as a valid option. |
| **TC03**: If user doesn’t check the box for agreeing the T&Cs. There should be an error message displayed and the system should prompt the user to check the box. |