# 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** | 500 | "low" |
| **TC02** | 1000 | “mid” |
| **TC03** | 2500 | “mid” |
| **TC04** | 3000 | “high” |
| **TC05** | 3500 | “high” |
| **TC06** | -1 | “error” |

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

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Input** | **Expected Output** |
| **TC07** | 999 | "low" |
| **TC08** | 1000 | “mid” |
| **TC09** | 1001 | “mid” |
| **TC10** | 2999 | “mid” |
| **TC11** | 3000 | “high” |
| **TC12** | 3001 | “high” |
| **TC13** | 0 | “low” |
| **TC14** | -1 | “error” |
| **TC15** | -1000 | “error” |
| **TC16** | -3000 | “error” |

## 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: 20** |

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

|  |
| --- |
| Shipping method **1** + **Yes** Gift + **Agree** Terms  Shipping method **2** + **No** Gift + **Don’t Agree** Terms  Shipping method **3** + **Agree** Terms + **No** Gift  Shipping method **4** + **Don't Agree** Terms + **Yes** Gift  Shipping method **5** + **Yes** Gift + **Agree** Terms  *Those would be the tests assuming the Place Order Button works as expected* |

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. |
| Example: No gift wrapper option selected or left blank. Either one of the radio buttons should be selected. |
| Example: Invalid shipping method, for example 6 |
| Example: Terms and Conditions unchecked. They should be required and checked. |