# Homework: Test Techniques

## 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** from the QA Basics course. You will find it the Resources.zip archive. The App categorizes the given **monthly income** into one of the following categories: "**low**", "**mid**", "**high"**. It works as follows:

* If the income is less than 1000.00, returns **"low"**
* If the income between 1000.00 and 2999.99 (inclusively), returns **"mid"**
* If the income is equal or bigger than 3000.00, 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 | 2333 | “mid” |
| TC03 | 3500 | “high” |
| TC04 | -900 | “error” |
| TC05 | -empty- | “Not accepted” |
| TC06 | 12A | “Not accepted” |

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

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Input** | **Expected Output** |
| TC07 | 999 | "low" |
| TC08 | **-0.1** | **“error”** |
| TC09 | **0** | **“low”** |
| TC10 | **1000.00** | **“mid”** |
| TC11 | **2999.99** | **“mid”** |
| TC12 | **3000.00** | **“high”** |
| TC13 | **3900.00** | **“high”** |
| TC14 | **“some text”** | **“error”** |
| TC15 | **Null** | **“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.

|  |
| --- |
| **Add a screenshot of the reduced test cases here**  **C:\Users\LENOVO\Pictures\Screenpresso\2025-06-08_20h48_12.png** |

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. |
| Drop – down menu value - 50 |
| Radio button – both on and off |
| Drop-down menu value = abc |