SE4452A Assignment 2

Demir Mensah – 250986865

**Q1.a Equivalence Class Testing**

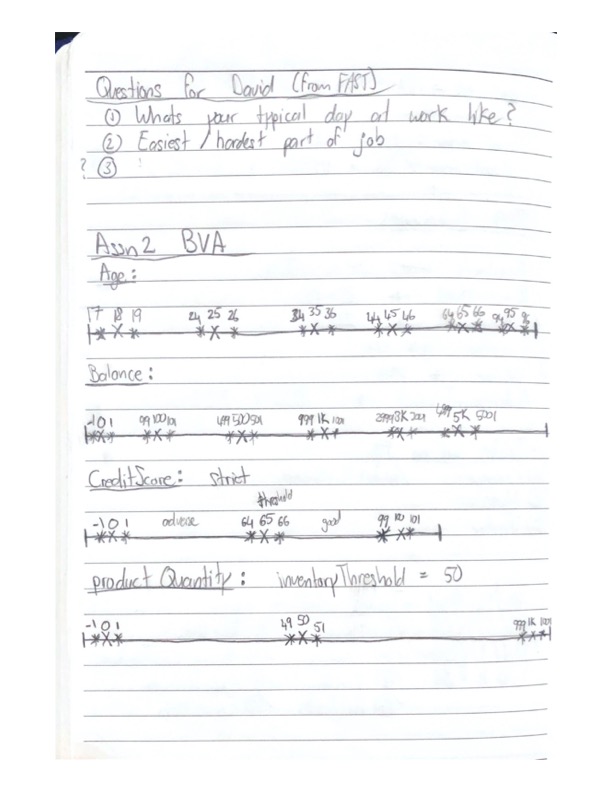
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AccountStatus** | | |  |  |  |
| **Age** | **Balance** | **ageFactor** | **balanceFactor** | **accountFactor** | **AccountStatus** |
| 15 | 0 | 0 | 0 | 0 | invalid |
| 20 | 50 | 10 | 5 | 50 | adverse |
| 30 | 250 | 15 | 15 | 225 | acceptable |
| 40 | 750 | 20 | 25 | 500 | acceptable |
| 20 | 2000 | 10 | 65 | 650 | good |
| 55 | 2000 | 45 | 65 | 2925 | excellent |
| 80 | 4000 | 20 | 150 | 3000 | excellent |
| 100 | 6000 | 0 | 0 | 0 | invalid |
| 100 | 750 | 0 | 25 | 0 | invalid |
| 40 | 6000 | 20 | 0 | 0 | invalid |

|  |  |  |  |
| --- | --- | --- | --- |
| **CreditStatus** |  |  |  |
| **creditScore** | **creditCheckMode** | **scoreThreshold** | **creditStatus** |
| -1 | strict | 65 | invalid |
| 150 | strict | 65 | invalid |
| 30 | strict | 65 | adverse |
| 80 | strict | 65 | good |
| -1 | default | 75 | invalid |
| 150 | default | 75 | invalid |
| 70 | default | 75 | adverse |
| 85 | default | 75 | good |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ProductStatus** | | q (0-1000 inc.) |  | invThreshold (0-1000 inc.) | |
| **product** | **productName** | **productQuantity** | **inventory** | **inventoryThreshold** | **productStatus** |
| shoes | shoes | 15 | [shoes,15] | -1 | invalid |
| shoes | shoes | 15 | [shoes,5] | 1200 | invalid |
| shoes | shoes | -2 | [shoes,-2] | 50 | invalid |
| shoes | shoes | 1500 | [shoes,1500] | 50 | invalid |
| shoes | shoes | 0 | [shoes,0] | 50 | soldout |
| shoes | shoes | 15 | [shoes, 15] | 50 | limited |
| shoes | shoes | 75 | [shoes, 75] | 50 | available |
| pencil | pencil | 50 | [shoes, 50] | n/a | invalid |

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderHandling** | | |  |
| 5 x 3 x 4 = 60 strong robust cases | | | |
| **accountStatus** | **creditStatus** | **productStatus** | **orderStatus** |
| excellent | adverse | soldout | accepted |
| excellent | adverse | limited | accepted |
| excellent | adverse | available | accepted |
| excellent | adverse | invalid | rejected |
| excellent | good | soldout | accepted |
| excellent | good | limited | accepted |
| excellent | good | available | accepted |
| excellent | good | invalid | rejected |
| excellent | invalid | soldout | rejected |
| excellent | invalid | limited | rejected |
| excellent | invalid | available | rejected |
| excellent | invalid | invalid | rejected |
| good | adverse | soldout | underReview |
| good | adverse | limited | underReview |
| good | adverse | available | underReview |
| good | adverse | invalid | rejected |
| good | good | soldout | accepted |
| good | good | limited | accepted |
| good | good | available | accepted |
| good | good | invalid | rejected |
| good | invalid | soldout | rejected |
| good | invalid | limited | rejected |
| good | invalid | available | rejected |
| good | invalid | invalid | rejected |
| acceptable | adverse | soldout | rejected |
| acceptable | adverse | limited | rejected |
| acceptable | adverse | available | underReview |
| acceptable | adverse | invalid | rejected |
| acceptable | good | soldout | pending |
| acceptable | good | limited | pending |
| acceptable | good | available | accepted |
| acceptable | good | invalid | rejected |
| acceptable | invalid | soldout | rejected |
| acceptable | invalid | limited | rejected |
| acceptable | invalid | available | rejected |
| acceptable | invalid | invalid | rejected |
| adverse | adverse | soldout | rejected |
| adverse | adverse | limited | rejected |
| adverse | adverse | available | rejected |
| adverse | adverse | invalid | rejected |
| adverse | good | soldout | rejected |
| adverse | good | limited | pending |
| adverse | good | available | underReview |
| adverse | good | invalid | rejected |
| adverse | invalid | soldout | rejected |
| adverse | invalid | limited | rejected |
| adverse | invalid | available | rejected |
| adverse | invalid | invalid | rejected |
| invalid | adverse | soldout | rejected |
| invalid | adverse | limited | rejected |
| invalid | adverse | available | rejected |
| invalid | adverse | invalid | rejected |
| invalid | good | soldout | rejected |
| invalid | good | limited | rejected |
| invalid | good | available | rejected |
| invalid | good | invalid | rejected |
| invalid | invalid | soldout | rejected |
| invalid | invalid | limited | rejected |
| invalid | invalid | available | rejected |
| invalid | invalid | invalid | rejected |

**Q1.b Boundary Value Analysis**

****

In the above screenshot, the boundaries for each variable in the purchase order code is shown. The boundaries are marked with an ‘X’ and the points close to the boundary are marked with ‘\*’ on the axis. After identifying the boundaries, I created the following tables of test cases in Excel. Note however that boundary value testing is not applicable to the order handling function as it takes strings as input and not integer values, hence we can’t test its boundaries.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Valid Age Intervals: [18, 25), [25, 35), [35, 45), [45, 65), [65, 95) | | | | |  |
| Valid Balance Intervals: [0, 100), [100, 500), [500, 1000), [1000, 3000), [3000, 5000) | | | | | |
|  |  |  |  |  |  |
| **AccountStatus** | |  |  |  |  |
| **Age** | **Balance** | **ageFactor** | **balanceFactor** | **accountFactor** | **AccountStatus** |
| 17 | -1 | 0 | 0 | 0 | invalid |
| 18 | 0 | 10 | 0 | 0 | invalid |
| 19 | 1 | 10 | 5 | 50 | adverse |
| 24 | 99 | 10 | 5 | 50 | adverse |
| 25 | 100 | 15 | 15 | 225 | acceptable |
| 26 | 101 | 15 | 15 | 225 | acceptable |
| 34 | 499 | 15 | 15 | 225 | acceptable |
| 35 | 500 | 20 | 25 | 500 | acceptable |
| 36 | 501 | 20 | 25 | 500 | acceptable |
| 44 | 999 | 20 | 25 | 500 | acceptable |
| 45 | 1000 | 45 | 65 | 2925 | excellent |
| 46 | 1001 | 45 | 65 | 2925 | excellent |
| 64 | 2999 | 45 | 65 | 2925 | excellent |
| 65 | 3000 | 25 | 150 | 3750 | excellent |
| 66 | 3001 | 25 | 150 | 3750 | excellent |
| 94 | 4999 | 25 | 150 | 3750 | excellent |
| 95 | 5000 | 0 | 0 | 0 | invalid |
| 96 | 5001 | 0 | 0 | 0 | invalid |

|  |  |  |  |
| --- | --- | --- | --- |
| **CreditStatus** | |  |  |
| **creditScore** | **creditCheckMode** | **scoreThreshold** | **creditStatus** |
| -1 | strict | 65 | invalid |
| 0 | strict | 65 | adverse |
| 1 | strict | 65 | adverse |
| 64 | strict | 65 | adverse |
| 65 | strict | 65 | good |
| 66 | strict | 65 | good |
| 99 | strict | 65 | good |
| 100 | strict | 65 | good |
| 101 | strict | 65 | invalid |
| -1 | default | 75 | invalid |
| 0 | default | 75 | adverse |
| 1 | default | 75 | adverse |
| 74 | default | 75 | adverse |
| 75 | default | 75 | good |
| 76 | default | 75 | good |
| 99 | default | 75 | good |
| 100 | default | 75 | good |
| 101 | default | 75 | invalid |

|  |  |  |  |
| --- | --- | --- | --- |
| **ProductStatus** | | q (0-1000 inc.) | invThreshold (0-1000 inc.) |
| **product** | **inventory** | **inventoryThreshold** | **productStatus** |
| shoes | [shoes,15] | -1 | invalid |
| shoes | [shoes,5] | 1200 | invalid |
| shoes | [shoes,-1] | 50 | invalid |
| shoes | [shoes,0] | 50 | soldout |
| shoes | [shoes,1] | 50 | limited |
| shoes | [shoes,49] | 50 | limited |
| shoes | [shoes,50] | 50 | available |
| shoes | [shoes,51] | 50 | available |
| shoes | [shoes, 999] | 50 | available |
| shoes | [shoes,1000] | 50 | available |
| shoes | [shoes,1001] | 50 | invalid |

**Q1.c Decision Table Testing**

**AccountStatus**

Age:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| R1 | 95 <= Age < 18 |
| R2 | 18 <= age < 25 |
| R3 | 25 <= age < 35 |
| R4 | 35 <= age < 45 |
| R5 | 45 <= age < 65 |
| R6 | 65 <= age < 95 |

Balance:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| B1 | 5000 <= balance <= 0 |
| B2 | 0 < balance < 100 |
| B3 | 100 <= balance < 500 |
| B4 | 500 <= balance < 1000 |
| B5 | 1000 <= balance < 3000 |
| B6 | 3000 <= balance < 5000 |

|  |  |
| --- | --- |
| **Actions** | **Output** |
| A1 | Invalid |
| A2 | Adverse |
| A3 | Acceptable |
| A4 | Good |
| A5 | Excellent |

Decision Table for accountStatus:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Rule 1 | Rule 2 | Rule 3 | Rule 4 | Rule 5 | Rule 6 | Rule 7 |
| **Conditions** | **Age** | R1 |  | R2 | R3 | R2 | R5 | R6 |
|  | **Balance** |  | B1 | B2 | B3 | B5 | B5 | B6 |
|  | **A1** | **X** | **X** |  |  |  |  |  |
|  | **A2** |  |  | **X** |  |  |  |  |
| **Actions** | **A3** |  |  |  | **X** |  |  |  |
|  | **A4** |  |  |  |  | **X** |  |  |
|  | **A5** |  |  |  |  |  | **X** |  |
|  | **A5** |  |  |  |  |  |  | **X** |

**CreditStatus**

creditScore:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| R1 | 100 <= creditScore < 0 |
| R2 | creditScore < scoreThreshold |
| R3 | creditScore >= scoreThreshold |

creditCheckMode, scoreThreshold:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| B1 | Strict, 65 |
| B2 | Default, 75 |

|  |  |
| --- | --- |
| **Actions** | **Output** |
| A1 | Invalid |
| A2 | Adverse |
| A3 | Good |

Decision Table for creditStatus function

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Rule 1 | Rule 2 | Rule 3 | Rule 4 | Rule 5 | Rule 6 |
| **Conditions** | **creditScore** | R1 | R2 | R3 | R1 | R2 | R3 |
|  | **creditCheckMode** | B1 | B1 | B1 | B2 | B2 | B2 |
|  | **A1** | **X** |  |  | **X** |  |  |
| **Actions** | **A2** |  | **X** |  |  | **X** |  |
|  | **A3** |  |  | **X** |  |  | **X** |

**ProductStatus**

productQuantity:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| R1 | productQuantity = 0 |
| R2 | productQuantity < inventoryThreshold |
| R3 | productQuantity >= inventoryThreshold |

|  |  |
| --- | --- |
| **Actions** | **Output** |
| A1 | Soldout |
| A2 | Limited |
| A3 | Available |

Decision Table for productStatus

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Rule 1 | Rule 2 | Rule 3 |
| **Conditions** | **productQuantity** | R1 | R2 | R3 |
|  | **A1** | **X** |  |  |
| **Actions** | **A2** |  | **X** |  |
|  | **A3** |  |  | **X** |

**OrderStatus Function**

accountStatus Rules:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| D1 | Invalid |
| D2 | Adverse |
| D3 | Acceptable |
| D4 | Good |
| D5 | Excellent |

creditStatus Rules:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| E1 | Invalid |
| E2 | Adverse |
| E3 | Good |

productStatus Rules:

|  |  |
| --- | --- |
| **Rule** | **Condition** |
| F1 | Soldout |
| F2 | Limited |
| F3 | Available |

|  |  |
| --- | --- |
| **Actions** | **Output** |
| A1 | Accepted |
| A2 | Pending |
| A3 | Under review |
| A4 | Rejected |

Decision Table for OrderHandling

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Rule 1 | Rule 2 | Rule 3 | Rule 4 | Rule 5 | Rule 6 | Rule 7 | Rule 8 |
| **Conditions** | **accountStatus** | D1 | D3 | D3 | D2 |  | D4 | D3 | D5 |
|  | **creditStatus** |  | E2 | E2 | E3 | E1 | E2 | E3 |  |
|  | **productStatus** |  | F1 | F2 | F1 |  |  | F2 |  |
|  | **A1** |  |  |  |  |  |  |  | **X** |
|  | **A2** |  |  |  |  |  |  | **X** |  |
| **Actions** | **A3** |  |  |  |  |  | **X** |  |  |
|  | **A4** | **X** | **X** | **X** | **X** | **X** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**Q1.d Coding Test Cases**

The test cases shown above were implemented in code and tested using Mocha. The source code is attached to the submission. The outputs of the equivalence class testing, boundary value testing, and decision table testing is below.

**Outputs of Equivalence Class Testing:**

The following is the analysis report of the equivalence class testing of the accountStatus, creditStatus, productStatus, and orderHandling functions.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

As you can see in the above screenshot, there were failing test cases when running the test cases for the productStatus function. After checking the code of the function, I noticed that there were errors present in the code. Below is a photo of the given productStatus function with the error on the right, and the fixed productStatus function on the right.

Text

Description automatically generatedText

Description automatically generated

Running the same test cases on the new productStatus function results in the correct output below:

Text

Description automatically generated

Running the test cases for the orderHandling function, we encountered many errors as seen below.

Graphical user interface

Description automatically generated

The original code given for the orderHandling function contains logical errors and is shown on the left below, and the corrected code is on the right:

Text

Description automatically generatedText

Description automatically generated

After fixing these errors, when the strong robust tests for the orderHandling function were run again, they all passed as seen on the output below:

A picture containing table

Description automatically generated

Now running all of the equivalence class tests at the same time, we obtain the following output which shows all 85 of the equivalence class tests passing:

A picture containing text

Description automatically generated

**Outputs of Boundary Value Testing:**

Text

Description automatically generatedRunning the test cases for the accountStatus function resulted in all cases passing as seen below:

Running the test cases for creditStatus function resulted in success as well:

A picture containing text

Description automatically generated

Finally, running the test cases for the productStatus function also resulted in all success:

Text

Description automatically generated

Please note that boundary value analysis is not applicable to the orderHandling function as it does not have integer conditional.

**Decision Table Testing**

Below are the outputs of the decision table tests:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

When running all of the test cases together, the following output shows all tests succeeding.

Graphical user interface, text

Description automatically generated