**Q1. DD-path graphs**

\*Please note: I have formatted the provided purchaseOrder2F20.js code using Visual Studio Code’s automatic formatting tool for legibility. If the line numbers don’t match up completely, this is why.

**getAgeFactor:**

|  |  |
| --- | --- |
| **Graph Node** | **Line in the code** |
| 1 | Lines 2 – 4 |
| 2 | Line 6 |
| 3 | Line 8 |
| 4 | Line 10 |
| 5 | Line 12 |
| 6 | Line 14 |
| 7 | Line 16 |
| 8 | Line 18 |
| 9 | Line 20 |
| 10 | Line 22 |
| 11 | Line 24 |
| 12 | Line 26 |
| 13 | Line 28 |

**A picture containing icon

Description automatically generated**

**getBalanceFactor:**

|  |  |
| --- | --- |
| **Graph Node** | **Line in the code** |
| 1 | Line 36 |
| 2 | Line 38 |
| 3 | Line 40 |
| 4 | Line 42 |
| 5 | Line 44 |
| 6 | Line 46 |
| 7 | Line 48 |
| 8 | Line 50 |
| 9 | Line 52 |
| 10 | Line 54 |
| 11 | Line 56 |
| 12 | Line 58 |
| 13 | Line 60 |

**A picture containing icon

Description automatically generated**

**accountStatus:**

|  |  |
| --- | --- |
| **Graph Node** | **Line in the code** |
| 1 | Lines 66 – 72 |
| 2 | Line 74 |
| 3 | Line 76 |
| 4 | Line 78 |
| 5 | Line 80 – 81 |
| 6 | Line 83 |
| 7 | Line 85 |
| 8 | Line 87 |
| 9 | Line 89 |
| 10 | Line 90 |

**A picture containing icon

Description automatically generated**

**creditStatus:**

|  |  |
| --- | --- |
| **Graph Node** | **Line in the code** |
| 1 | Line 98 |
| 2 | Line 100 |
| 3 | Line 103 |
| 4 | Line 105 |
| 5 | Line 107 |
| 6 | Line 109 |
| 7 | Line 111 |
| 8 | Line 113 |
| 9 | Line 115 |

**A picture containing text, pool ball, night sky

Description automatically generated**

**productStatus:**

|  |  |
| --- | --- |
| **Graph Node** | **Line in the code** |
| 1 | Lines 120 - 122 |
| 2 | Line 122 i <= […] |
| 3 | Line 123 |
| 4 | Line 122 i++ |
| 5 | Lines 124 - 126 |
| 6 | Line 128 |
| 7 | Line 130 |
| 8 | Line 132 |
| 9 | Line 134 |
| 10 | Line 137 |

**A picture containing icon

Description automatically generated**

**orderHandling:**

|  |  |
| --- | --- |
| **Graph Node** | **Line in the code** |
| 1 | Lines 143 - 149 |
| 2 | Line 153 |
| 3 | Line 155 |
| 4 | Line 157 |
| 5 | Line 160 |
| 6 | Line 162 |
| 7 | Line 164 |
| 8 | Line 166 |
| 9 | Unwritten; “return NULL” |

**A picture containing text, pool ball, vector graphics

Description automatically generated**

**Q2. Basis paths**

**getAgeFactor and getBalanceFactor:**

When looking at the DD-paths, it is evident these functions are functionally equivalent. Thus, there are 6 paths needed to test every node at least once. The paths are:

start – 1 – 2 – 13 – end

start – 1 – 3 – 4 – 13 – end

start – 1 – 3 – 5 – 6 – 13 – end

start – 1 – 3 – 5 – 7 – 8 – 13 – end

start – 1 – 3 – 5 – 7 – 9 – 10 – 13 – end

start – 1 – 3 – 5 – 7 – 9 – 11 – 12 – 13 – end

**accountStatus:**

There are 5 paths needed:

start – 1 – 2 – end

start – 1 – 3 – 4 – end

start – 1 – 3 – 5 – 6 – end

start – 1 – 3 – 5 – 7 – 8 – end

start – 1 – 3 – 5 – 7 – 9 – 10 – end

**creditStatus:**

There are 3 paths needed:

start – 1 – 2 – 3 – 4 – 7 – 8 – end

start – 1 – 2 – 3 – 5 – 7 – 9 – end

start – 1 – 2 – 3 – 5 – 6 – 7 – 8 – end

**productStatus:**

There are 4 paths needed to hit every node. I have also built these paths for loop testing too:

start – 1 – 2 – 10 – end (skip loop)

start – 1 – 2 – 3 – 5 – 6 – end (zero loops)

start – 1 – 2 – 3 – 4 – 2 – 3 – 5 – 7 – 8 – end (one loop)

start – 1 – 2 – 3 – 4 – 2 – 3 – 4 – 2 – 3 – 5 – 7 – 9 – end (multiple loops)

**orderHandling:**

There are technically 5 paths needed to hit every node, however the path that hits node 9 is invalid, as there is no code written to handle a last false, so the function does not return anything. As such, the path does not need to be tested for, so there are only 4 paths to test:

start – 1 – 2 – end

start – 1 – 2 – 3 – 4 – end

start – 1 – 2 – 3 – 5 – 6 – end

start – 1 – 2 – 3 – 5 – 7 – 8 – end

**Q3. Structural Unit Testing**

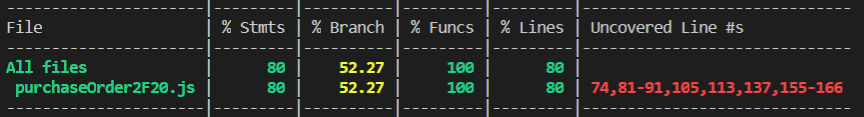
1: This is achieved because the tests hit every node in the code.

2: This is achieved because the tests hit every edge they can in the code.

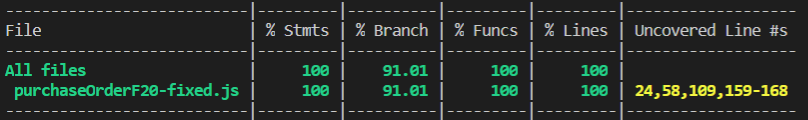
3: “Path” not well defined, but by hitting every node and edge, every path is covered (see my paths in the above Q.2. and in the attached test.js code)

**Q4. Istanbul**

Note: Istanbul is now “nyc.js”. The below screenshot is the tests run on the original unmodified code (save for fixing that one syntax error!) provided for this lab- due to the errors there is a lot of code that is missed by the tests.



When these new tests are run on my fixed code purchaseOrderF20-fixed.js I submitted as part of assignment 2, the results are better:



The %Branch is still not 100, however, when the uncovered lines are inspected, they are “else ifs” that are logically impossible to reach, or they are outside the scope of the testing as detailed in the assignment manual (i.e. F6 orderHandling treat the multi-condition decision (if statement) as a single node). When these are accounted for, the %Branch is as good as it can get.

**Bonus: Q5. MM-Path Testing**

The module execution paths are mostly the same as the Q2. basis paths above, but there are 2 changes, to accountStatus and orderHandling to account for the internal function calls. The MEP paths are as follows:

**getAgeFactor:**

1. start – 1 – 2 – 13 – end (0)
2. start – 1 – 3 – 4 – 13 – end (10)
3. start – 1 – 3 – 5 – 6 – 13 – end (15)
4. start – 1 – 3 – 5 – 7 – 8 – 13 – end (20)
5. start – 1 – 3 – 5 – 7 – 9 – 10 – 13 – end (45)
6. start – 1 – 3 – 5 – 7 – 9 – 11 – 12 – 13 – end (25)

**getBalanceFactor:**

1. start – 1 – 2 – 13 – end (0)
2. start – 1 – 3 – 4 – 13 – end (5)
3. start – 1 – 3 – 5 – 6 – 13 – end (15)
4. start – 1 – 3 – 5 – 7 – 8 – 13 – end (25)
5. start – 1 – 3 – 5 – 7 – 9 – 10 – 13 – end (65)
6. start – 1 – 3 – 5 – 7 – 9 – 11 – 12 – 13 – end (120)

**accountStatus:**

1. start – function call
2. function return – 1 – 2 – end (invalid)
3. function return – 1 – 3 – 4 – end (adverse)
4. function return – 1 – 3 – 5 – 6 – end (acceptable)
5. function return – 1 – 3 – 5 – 7 – 8 – end (good)
6. function return – 1 – 3 – 5 – 7 – 9 – 10 – end (excellent)

**creditStatus:**

1. start – 1 – 2 – 3 – 4 – 7 – 8 – end (invalid)
2. start – 1 – 2 – 3 – 5 – 7 – 9 – end (adverse)
3. start – 1 – 2 – 3 – 5 – 6 – 7 – 8 – end (good)

**productStatus:**

1. start – 1 – 2 – 10 – end (invalid)
2. start – 1 – 2 – 3 – 5 – 6 – end (soldout)
3. start – 1 – 2 – 3 – 4 – 2 – 3 – 5 – 7 – 8 – end (limited)
4. start – 1 – 2 – 3 – 4 – 2 – 3 – 4 – 2 – 3 – 5 – 7 – 9 – end (available)

**orderHandling:**

1. start – function call
2. function return – 2 – end (rejected)
3. function return – 3 – 4 – end (accepted)
4. function return – 3 – 5 – 6 – end (underReview)
5. function return – 3 – 5 – 7 – 8 – end (pending)

**Minimum set of paths needed to hit every MEP:**

There will need to be 6 paths to cover every MEP in the getAgeFactor and getBalanceFactor functions:

1. **Path1 (returns “rejected”):** orderhandling-1 -> accountStatus-1 -> getAgeFactor-1 -> getBalanceFactor-1 -> accountStatus-2 -> creditStatus-1 -> productStatus-1 -> orderhandling-2
2. **Path2 (returns “accepted”):** orderhandling-1 -> accountStatus-1 -> getAgeFactor-5 -> getBalanceFactor-6 -> accountStatus-6 -> creditStatus-3 -> productStatus-4-> orderhandling-3
3. **Path3 (returns “underReview”):** orderhandling-1 -> accountStatus-1 -> getAgeFactor-4 -> getBalanceFactor-3 -> accountStatus-4 -> creditStatus-2 -> productStatus-4 -> orderhandling-4
4. **Path4 (returns “pending”):** orderhandling-1 -> accountStatus-1 -> getAgeFactor-3 -> getBalanceFactor-4 -> accountStatus-4 -> creditStatus-3 -> productStatus-3 -> orderhandling-5
5. **Path5 (returns “rejected”):** orderhandling-1 -> accountStatus-1 -> getAgeFactor-6 -> getBalanceFactor-2 -> accountStatus-3 -> creditStatus-1 -> productStatus-1 -> orderhandling-2
6. **Path6 (returns “rejected”):** orderhandling-1 -> accountStatus-1 -> getAgeFactor-2 -> getBalanceFactor-5 -> accountStatus-5 -> creditStatus-1 -> productStatus-2 -> orderhandling-2

**Results:**

Please fine the code in the test.js file in the .zip. Below are the test results for the 6 MM-Path tests:

