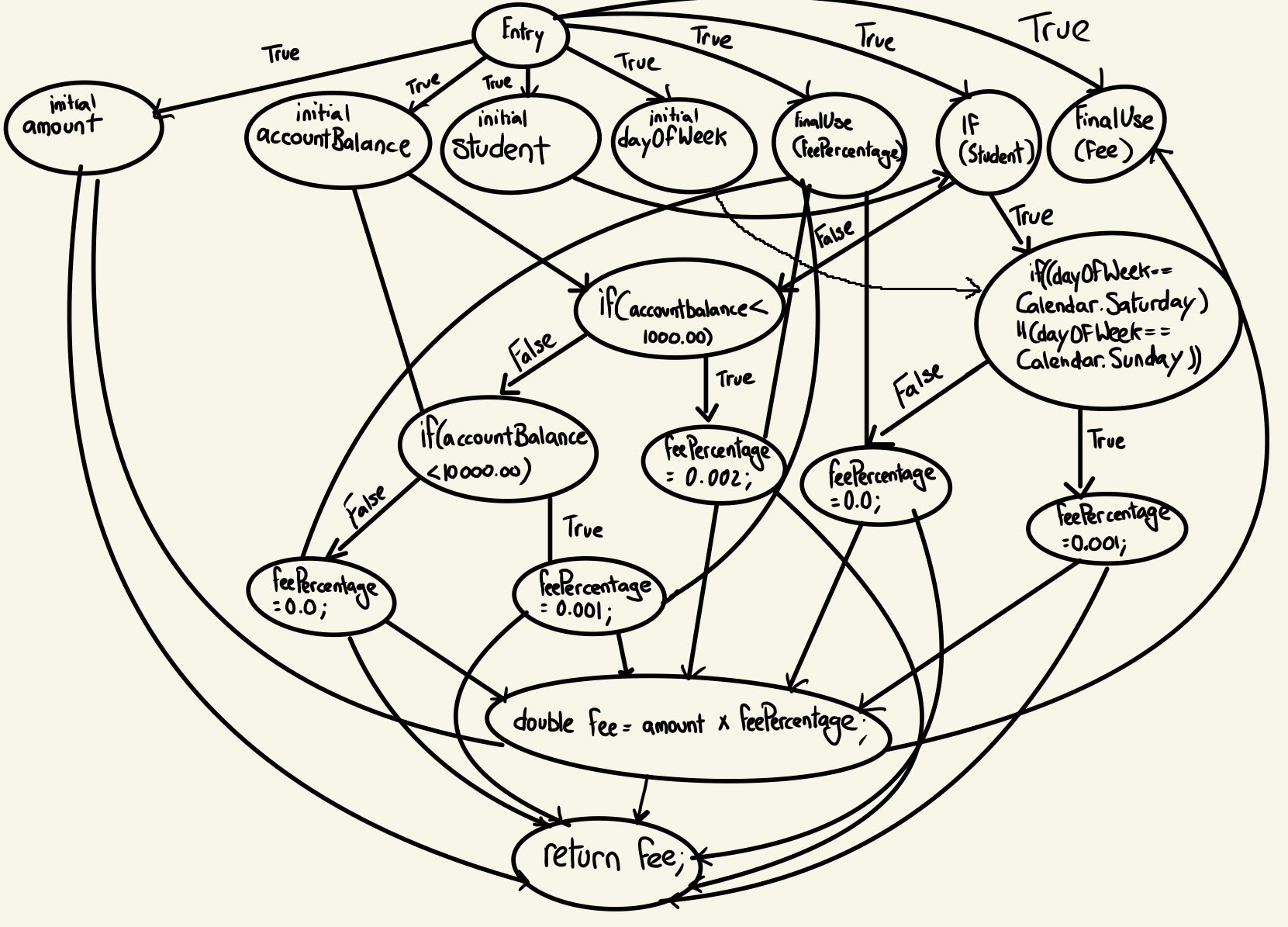
**Assignment 2 - Whitebox Testing**

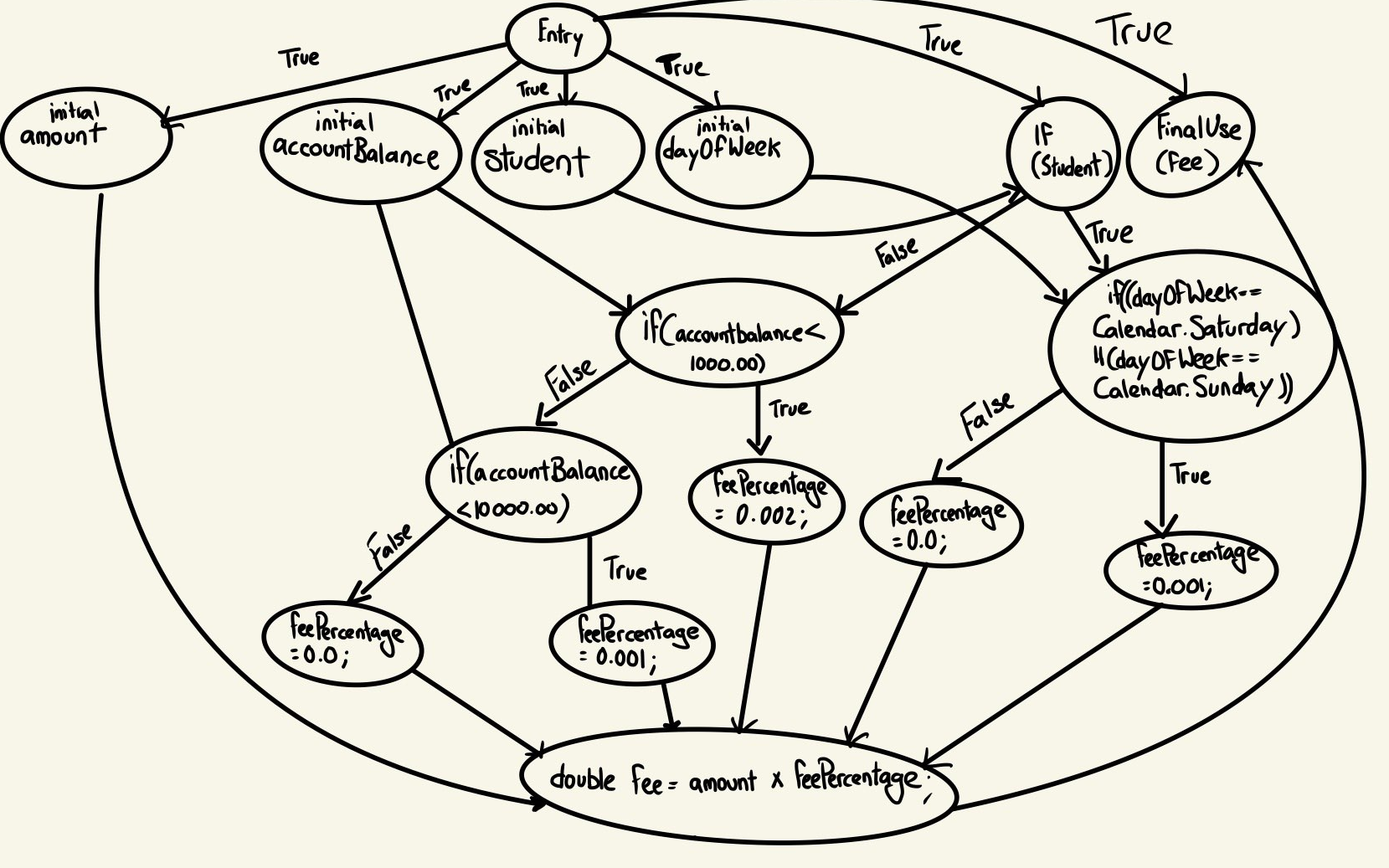
**Withdrawal**

As a result of the assignment asking us to calculate the slice based on FinalUse(fee) - the statement return fee we know that we are looking for a ***backward slice***. As a result, we put together a PDG (Program Dependency Graph) and used that to trace from finalUse(fee) and give us our final graph after calculating the slice.

**Initial PDG**

****

**Computed Slice**

****

**Test Cases:**

As a result of the above graph, we had calculated 6 different test cases to cover the various statements in the withdrawal operation.

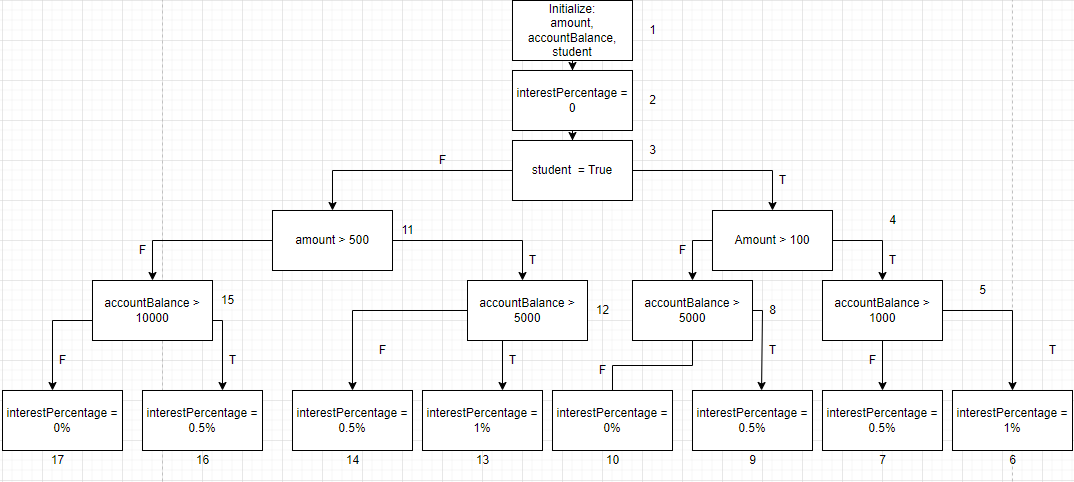
| amount | accountBalance | student | dayOfWeek | Result | expectedResult | Pass/Fail |
| --- | --- | --- | --- | --- | --- | --- |
| 100 | 1000 | true | 7 (Saturday) | 0.0 | 0.0 | Pass |
| 100 | 1000 | true | 1 (Sunday) | 0.0 | 0.0 | Pass |
| 100 | 1000 | true | 5 (Friday) | 0.1 | 0.1 | Pass |
| 100 | 999 | false | 5 (Friday) | 0.2 | 0.2 | Pass |
| 100 | 9999 | false | 5 (Friday) | 0.1 | 0.1 | Pass |
| 100 | 99999 | false | 5 (Friday) | 0 | 0 | Pass |

As a result, every test passed. The initial 2 tests were made to check if the statement if ((dayOfWeek == Calendar.*SATURDAY*) || (dayOfWeek == Calendar.*SUNDAY*)) was functioning properly. After that, in order to check if the else statement was functioning as expected, we tested its results. The fourth test case was to test if if (accountBalance < 1000.00) was functioning as expected. The fifth test case was to test if the next statement else if (accountBalance < 10000.00) worked as intended. The final test case was to ensure that the else{ statement in line 22 of feesCalculator was working as intended.

**Deposit**

For deposit interest testing for this assignment. We were supposed to calculate the DU paths and create the corresponding unit test coverage for each path using white box testing methods. As such I charted out every path using a flow chart, and then calculated the paths and created the corresponding 8 test cases for the 8 Paths.

**DU Paths**

****

**List of Paths**

1. 1-2-3-4-5-6
2. 1-2-3-4-5-7
3. 1-2-3-4-8-9
4. 1-2-3-4-8-10
5. 1-2-3-11-12-13
6. 1-2-3-11-12-14
7. 1-2-3-11-15-16
8. 1-2-3-11-15-17

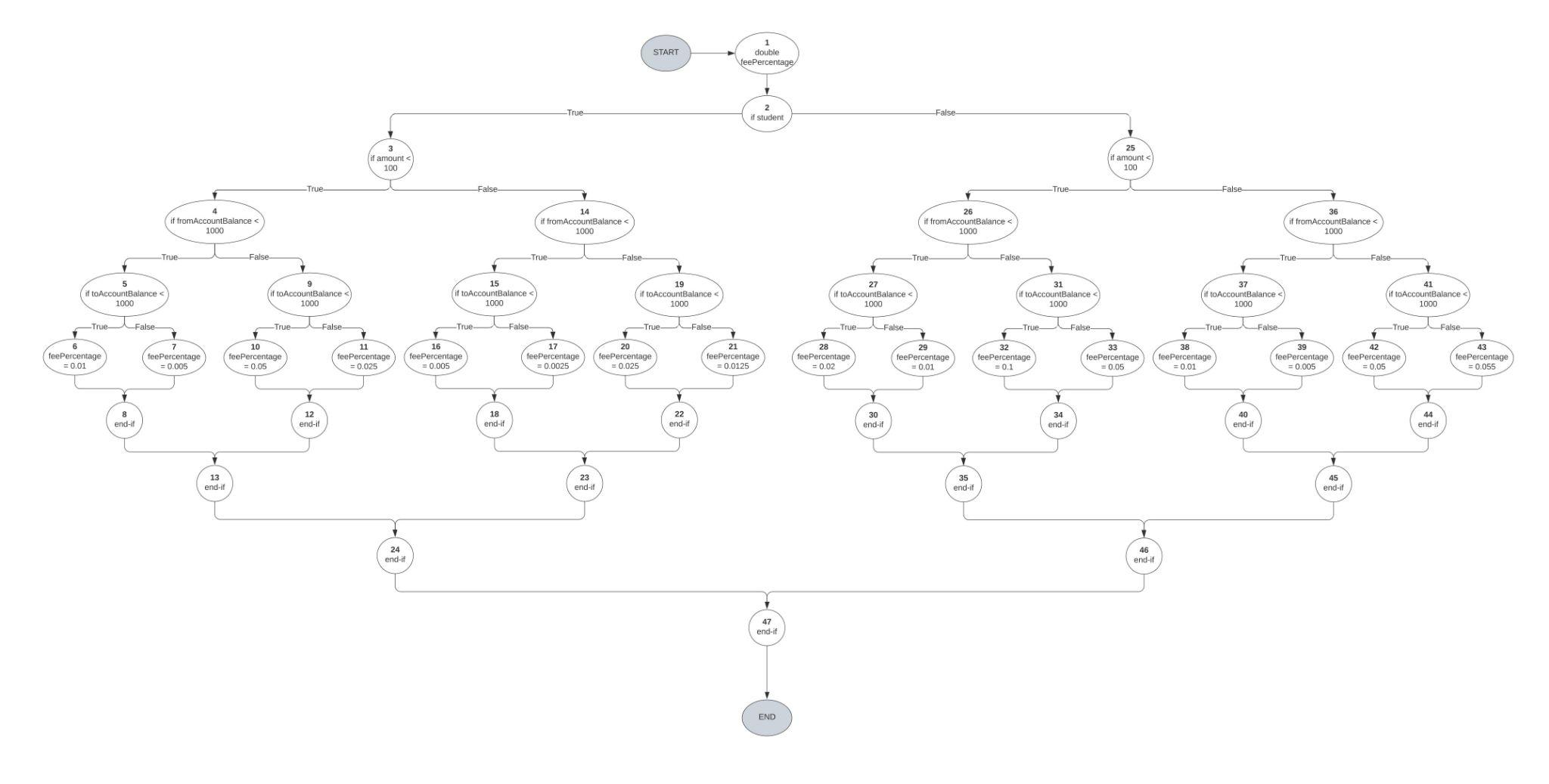
**Test Case Results**

| **Test Case** | **Input** | **Path** | **Expected** | **Actual** | **Result** |
| --- | --- | --- | --- | --- | --- |
| T1 | 1000 ,10000, true | 1-2-3-4-5-6 | 1% | 1% | **Pass** |
| T2 | 1000 ,100, true | 1-2-3-4-5-7 | 0.5% | 0.5% | **Pass** |
| T3 | 100 ,10000, true | 1-2-3-4-8-9 | 0.5% | 0.5% | **Pass** |
| T4 | 100 ,1000, true | 1-2-3-4-8-10 | 0% | 0.2% | **Fail** |
| T5 | 1000 ,10000, false | 1-2-3-11-12-13 | 1% | 1% | **Pass** |
| T6 | 1000 ,1000, false | 1-2-3-11-12-14 | 0.5% | 0.5% | **Pass** |
| T7 | 100 ,100000, false | 1-2-3-11-15-16 | 0.5% | 0.5% | **Pass** |
| T8 | 100 ,1000, false | 1-2-3-11-15-17 | 0% | 0% | **Pass** |

**Transfer**

For the transfer interest testing, we first made the control flow graph using the assignment code and came out with the graph given below. By using the graph, we calculated the cyclomatic complexity, which resulted in us needing 16 independent basis paths. The table below shows the input we used, the paths we took, and finally, the results.

**Basis Path Testing Flow Chart:**



**Test Cases Needed:**

Cyclomatic Complexity = # edges - # nodes + 2  
= 63 - 49 + 2  
= 16 total test cases

or

Cyclomatic Complexity = # enclosed areas + 1  
= 15 + 1  
= 16 total test cases

**Basis Path Independent Test Paths:**

| **Test Cases** | **Input** | **Independent Path Covered** | **Result** |
| --- | --- | --- | --- |
| T1 | 10, 100, 100, true | 1, 2, 3, 4, 5, 6, 8, 13, 24, 47 | Fail |
| T2 | 10, 100, 10000, true | 1, 2, 3, 4, 5, 7, 8, 13, 24, 47 | Fail |
| T3 | 10, 10000, 100, true | 1, 2, 3, 4, 9, 10, 12, 13, 24, 47 | Fail |
| T4 | 10, 10000, 10000, true | 1, 2, 3, 4, 9, 11, 12, 13, 24, 47 | Fail |
| T5 | 1000, 100, 100, true | 1, 2, 3, 14, 15, 16, 18, 23, 24, 47 | Fail |
| T6 | 1000, 100, 10000, true | 1, 2, 3, 14, 15, 17, 18, 23, 24, 47 | Fail |
| T7 | 1000, 10000, 100, true | 1, 2, 3, 14, 19, 20, 22, 23, 24, 47 | Fail |
| T8 | 1000, 10000, 10000, true | 1, 2, 3, 14, 19, 21, 22, 23, 24, 47 | Fail |
| T9 | 10, 100, 100, false | 1, 2, 25, 26, 27, 28, 30, 35, 46, 47 | Fail |
| T10 | 10, 100, 10000, false | 1, 2, 25, 26, 27, 29, 30, 35, 46, 47 | Fail |
| T11 | 10, 10000, 100, false | 1, 2, 25, 26, 31, 32, 34, 35, 46, 47 | Fail |
| T12 | 10, 10000, 10000, false | 1, 2, 25, 26, 31, 33, 34, 35, 46, 47 | Fail |
| T13 | 1000, 100, 100, false | 1, 2, 25, 36, 37, 38, 40, 45, 46, 47 | Fail |
| T14 | 1000, 100, 10000, false | 1, 2, 25, 36, 37, 39, 40, 45, 46, 47 | Fail |
| T15 | 1000, 10000, 100, false | 1, 2, 25, 36, 41, 42, 44, 45, 46, 47 | Fail |
| T16 | 1000, 10000, 10000, false | 1, 2, 25, 36, 41, 43, 44, 45, 46, 47 | Fail |