# Test Description 1

**Test Name or ID**:

**Test Type**: Black box

**Description**: Testing functionality of accepting only valid packages

**Test Function**: validatePackage()

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T001 | Program should accept normal weight | p.weight = 100 | 1 | 1 | Pass |
| T002 | Program should reject weight over 1200 | p.weight = 1201 | 0 | 0 | Pass |
| T003 | Program should accept sizes: 0.5, 1, or 5 | p.size = 1 | 1 | 1 | Pass |
| T004 | Program should reject any other package sizes | p.size = 3 | 0 | 0 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Test Description 2

**Test Name or ID**:

**Test Type**: Blackbox

**Description**: Testing functionality of determining which truck is less full

**Setup:** We are going to use VS2022 to compile the code and prepare test data, implement test case and execute the test.

**Test Function**: int truckLessFull(struct Truck\* firstTruck, struct Truck\* secondTruck);

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T005 | Equal current weight (assume both max capacity) | firstTruck->currentWeight = 100  secondTruck-> currentWeight= 100 | -1 |  |  |
| T006 | First truck has more available weight (assume both at max capacity) | firstTruck->currentWeight = 0  secondTruck->currentWeight= 50 | 1 |  |  |
| T007 | Second truck has more available weight (assume both at max capacity) | firstTruck->currentWeight = 500  secondTruck->currentWeight= 50 | 0 |  |  |
| T008 | Neither has a limiting factor but first truck has a higher percentage of space available | firstTruck->currentWeight =500  firstTruck->currentCapacity=20  secondTruck->currentWeight =600  secondTruck->currentCapacity=25 | 1 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Test Description 3

**Test Name or ID**:

**Test Type**: Black box

**Description**: Testing for destination address valid or not

**Setup:** We are going to use VS2022 to compile the code and prepare test data, implement test case and execute the test.

**Test Function**: int validDestination(struct map\* maps, int row, char col)

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T009 | Testing for destination out of columns | row = 8 col = ’Z’ | 0 |  |  |
| T010 | Testing for destination out of rows | row = 26 col = ’A’ | 0 |  |  |
| T011 | Testing for black location within map | row = 13 col = ’J’ | 1 |  |  |
| T012 | Testing for white location within map | row = 25 col = ’Y’ | 0 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Test Description 4

**Test Name or ID**:

**Test Type**: Black box

**Description**: Compare the lengths of two routes and returns the correct result

**Setup:** Ensure the struct Route are defined and available.

**Test Function**: int compareRouteLengths(struct Route\* route1, struct Route\* route2)

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T013 | Route1 shorter than Route2 | route1.numPoints = 20, route2.numPoints = 50 | -1 |  |  |
| T014 | Route2 shorter than Route1 | route1.numPoints = 50, route2.numPoints = 20 | 1 |  |  |
| T015 | Route1 and Route2 are equal length | route1.numPoints = 10, route2.numPoints = 10 | 0 |  |  |
| T016 | Both routes are empty | route1.numPoints = 0, route2.numPoints = 0 | 0 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Test Description 5 (Given function)

**Test Name or ID**:

**Test Type**: Black box

**Description**: This function is testing for equal point

**Setup:** We are going to use VS2022 to compile the code and prepare test data, implement test case and execute the test.

**Test Function**: int eqPt(const struct Point p1, const struct Point p2)

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T017 | Testing for p1 equal to p2 | p1.row = ‘7’ p1.col = ‘C’ p2.row = ‘7’ p2.col = ‘C’ | 1 |  |  |
| T018 | Testing for p1.row != p2.row and p1.col == p2.col | p1.row = ‘23’ p1.col = ‘C’ p2.row = ‘7’ p2.col = ‘C’ | 0 |  |  |
| T019 | Testing for p1.row == p2.row and p1.col != p2.col | p1.row = ‘7’ p1.col = ‘C’ p2.row = ‘7’ p2.col = ‘X’ | 0 |  |  |
| T020 | Testing for p1.row != p2.row and p1.col != p2.col | p1.row = ‘9’ p1.col = ‘C’ p2.row = ‘14’ p2.col = ‘D’ | 0 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Test Description 6 (Given function)

**Test Name or ID**:

**Test Type**: Black box

**Description**: Testing if row number can be correctly returned

**Test Function**: getNumRows()

**Setup:** We are going to use VS2022 to compile the code and prepare test data, implement test case and execute the test.

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T021 | Testing a normal Map | Map.numRows = 5 | 5 | 5 | Pass |
| T022 | Testing with edge cases | Map.numRows = 0 | 0 | 0 | Pass |
| T023 | Testing for a map that did not initiate with data | Map.numRows | Undefined behavior | Undefined behavior | Fail |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Program might produce unexpected behavior if the map parameter contains undefined value. Fixed program to return 0 if map is in invalid state.

# Test Description 7

**Test Name or ID**:

**Test Type**: **WHITE BOX**

**Description**: Testing functionality of determining which truck is less full

**Setup:** We are going to use VS2022 to compile the code and prepare test data, implement test case and execute the test.

**Test Function**: int truckLessFull(struct Truck\* firstTruck, struct Truck\* secondTruck);

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T024 | Truck 1 greater current weight, less current capacity, but truck 2 max capacity | firstTruck->currentWeight = 100  secondTruck-> currentWeight= 50  firstTruck->currentCapacity = 20  secondTruck->currentCapacity = 50 (full) | 1 | 1 | Pass |
| T025 | Truck 2 greater current weight, less current capacity, but truck 1 max capacity | firstTruck->currentWeight = 50  secondTruck-> currentWeight= 100  firstTruck->currentCapacity = 50 (full)  secondTruck->currentCapacity = 20 | 0 | 0 | Pass |
| T026 | Truck 1 greater current capacity, less current weight, but truck 2 max weight | firstTruck->currentWeight = 100  secondTruck-> currentWeight= 1200 (max)  firstTruck->currentCapacity = 40  secondTruck->currentCapacity = 20 | 1 | 1 | Pass |
| T027 | Truck 2 greater current capacity, less current weight, but truck 1 max weight | firstTruck->currentWeight = 1200 (max)  secondTruck-> currentWeight= 100  firstTruck->currentCapacity = 20  secondTruck->currentCapacity = 40 | 0 | 0 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Test Description 8

**Test Name or ID**:

**Test Type**: WHITEBOX

**Description**: Testing functionality of accepting only valid packages

**Test Function**: validatePackage()

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T028 | Program should accept edge weight 1200 | p.weight = 1200 | 1 | 1 | Pass |
| T029 | Program should reject weight of 0 | p.weight = 0 | 0 | 1 | Fail (pass after fix) |
| T030 | Program should reject sizes other than: 0.5, 1, or 5 | p.size = 1.1 | 0 | 0 | Pass |
| T031 | Program should reject sizes other than: 0.5, 1, or 5 | p.size = 4.9 | 0 | 0 | Pass |
| T032 | Program should reject negative weight | p.weight = -1 | 0 | 0 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Found program to allow 0 as package weight but in reality packages cannot be weighed as 0. Bug reported & fixed by Doris

# Test Description 9 (Given function)

**Test Name or ID**:

**Test Type**: WHITEBOX

**Description**: Testing if row number can be correctly returned

**Test Function**: getNumRows()

**Setup:** We are going to use VS2022 to compile the code and prepare test data, implement test case and execute the test.

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T033 | Testing a invalid Map data | Map.numRows = -5 | 0 | -5 | Pass |
| T034 | Testing large map | Map.numRows = 1000 | 1000 | 1000 | Pass |
| T035 | Testing invalid map | Map\* = nullptr | 0 | Undefined behavior | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Program might produce unexpected behavior if the map parameter contains undefined value. Fixed program to return 0 if map is in invalid state.

# Test Description 10

**Test Name or ID**:

**Test Type**: WHITEBOX

**Description**: Testing functionality of accepting only valid destinations

**Test Function**: int validDestination(struct map\* maps, int row, char col)

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T036 | Test with an invalid destination | row = 1 col = ‘A’ | 0 | 0 | Pass |
| T037 | Test with a valid destination | row = 1 col = ‘E’ | 1 | 1 | Pass |
| T038 | Test with row out of bounds | row = -1 col = ‘A’ | 0 | 0 | Pass |
| T039 | Test with column out of bounds | row = 2 col = ‘Z’ | 0 | 0 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Integration Test - Test Description 11

**Test Name or ID**:

**Test Type**: Integration of isTruckFull and addPackageToTruck

**Description**: Testing the integration of two functions to ensure foolproof entry of adding a package to truck

**Test Function**: isTruckFull which is invoked from addPackageToTruck

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T040 | Tests if package will be added to truck if it is above max capacity i | truck.currentCapacity = 60;  truck.currentWeight = 1300;  pkg.size = 5;  pkg.weight = 50; | 0 | 0 | Pass |
| T041 | Tests if package will be added to truck if it is below max capacity | truck.currentCapacity = 0;  truck.currentWeight = 0;  pkg.size = 5;  pkg.weight = 50; | 1 | 1 | Pass |
| T042 | Tests if a truck will reject adding a package if it will overflow the truck | truck.currentCapacity = 49;  truck.currentWeight = 1160;  pkg.size = 5;  pkg.weight = 50; | 0 | 0 | Pass |
| T043 | Tests if a package will be added to truck if it ends up being equal to max capacity | truck.currentCapacity = 0;  truck.currentWeight = 0;  pkg.size = 50;  pkg.weight = 1200; | 1 | 1 | 0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Integration Test - Test Description 12

**Test Name or ID**: integration test of validating user input

**Test Type**: Integration test

**Description**: Testing the integration of two functions to ensure foolproof entry of package and destination specifications

**Test Function**: validatePackage and validateDestination are invoked from validInput

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T044 | Tests if user enters invalid weight | //Valid  pkg.size = 5;  //Invalid  pkg.weight = 1300;  //Valid  int row = 8;  //Valid  char col = 'Y'; | 0 | 0 | Pass |
| T045 | Tests of user enters invalid size | //Valid  pkg.size = 5;  //Valid  pkg.weight = 500;  //Valid  int row = 8;  //Invalid  char col = 'Z'; | 0 | 0 | Pass |
| T046 | Tests if user enters invalid column | //Valid  pkg.size = 5;  //Valid  pkg.weight = 500;  //Valid  int row = 8;  //Invalid  char col = 'Z'; | 0 | 0 | Pass |
| T047 | Tests if user enters invalid row | //Valid  pkg.size = 5;  //Valid  pkg.weight = 500;  //Invalid  int row = -1;  //Valid  char col = 'Y'; | 0 | 0 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Integration Test - Test Description 13

**Test Name or ID**: integration test of adding the package to the selection truck

**Test Type**: Integration test

**Description**: Testing the integration of two functions to ensure foolproof entry of adding a package to the selected truck

**Test Function**: addPackageToTruck is called by addPackageToSelectedTruck

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T048 | Tests if package will be added to the selected truck which has room for it | truck.currentCapacity = 0;  truck.currentWeight = 0;  pkg.size = 1;  pkg.weight = 10; | 1 | 1 | Pass |
| T049 | Tests if package will be added to the selected truck which is maxed out on weight | truck.currentCapacity = 20;  truck.currentWeight = 1200;  pkg.size = 5;  pkg.weight = 10; | 0 | 0 | Pass |
| T050 | Tests if package will be added to the selected truck which is maxed out on capacity | truck.currentCapacity = 50;  truck.currentWeight = 20;  pkg.size = 5;  pkg.weight = 10; | 0 | 0 | Pass |
| T051 | Tests if a package will be added to selected truck if it ends up being equal to max capacity | truck.currentCapacity = 1190;  truck.currentWeight = 0=45;  pkg.size = 5;  pkg.weight = 10; | 1 | 1 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.

# Integration Test - Test Description 14

**Test Name or ID**

**Test Type**: Integration test

**Description**: Testing the integration of the two functions

**Test Function**: truckIsFull is called by truckLessFull

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TestID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| T052 | Truck 1 is full | t1.currentCapacity = MAX\_CAPACITY; | 0 | 0 | Pass |
| T053 | Truck 2 is full | T2.currentCapacity = MAX\_CAPACITY; | 1 | 1 | Pass |
| T054 | Both trucks are empty | t1.currentCapacity =0;  t2.currentCapacity =0; | -1 | -1 | Pass |
| T055 | Both trucks are full | t1.currentCapacity = MAX\_CAPACITY;  t2.currentCapacity = MAX\_CAPACITY; | -1 | -1 | Pass |
|  | Omg we did it |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

Description of each bug found above and how to reproduce it.