# Test Description

**Test Name or ID**: isValidDest

**Test Type**: Black box

**Description**: we are testing the function “isValidDest” to ensure that the valid input of a shipment. Because the validity of the shipping area affects the proper shipment. We have normal, boundary point, negative value, and exceeding value test cases.

**Setup:** Visual Studio native unit test framework

**Test Function**: isValidDest - check if the field of the given shipment is within valid value.

**Test Scenarios: see the chart below**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0001 | Test valid point  Valid box size | 10,10 | True/1 | 1 | pass |
| TS0002 | Test valid boundary point | 24,24 | True/1 | 1 | pass |
| TS0003 | Test invalid row with negative value | -1,10 | false/0 | 0 | pass |
| TS0004 | Test invalid column with exceeding value | 10,25 | false/0 | 0 | pass |
| TS0005 | Test invalid column with negative value | 10, -1 | false/0 | 0 | pass |
| TS0006 | Test invalid row with exceeding value | 25,10 | false/0 | 0 | pass |
|  |  |  |  |  |  |

**Bugs Found**:

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

**Test Name or ID**: limitingFactorWithShipment

**Test Type**: Black box

**Description**: When two trucks are the same distance away from the destination, we should put the package in the truck which is less full using limiting factor percentage. Optimize all trucks usage. Function `limitingFactorWithShipment` calculates the limiting factor of a truck with an extra shipment and returns the percentage. We have normal, an empty truck, a full load truck and over capacity test cases.

**Setup:** Visual Studio native unit test

**Test Function**: limitingFactorWithShipment - calculate the limiting factor of a truck with an extra shipment

**Test Scenarios: see the chart below (max weight is 2500kg max volume is 100 cubic meters)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0007 | Test return weight percentage | (2400, 50) (100, 10) | 100% | 100% | pass |
| TS0008 | Test return volume percentage | (2000,90) (100,10) | 100% | 100% | pass |
| TS0009 | Test both equal | (2450,95) (50,5) | 100% | 100% | pass |
| TS0010 | Test no additional weight | (2300,80) (0,20) | 100% | 100% | pass |
| TS0011 | Test no addition volume | (2200,90) (300,0) | 100% | 100% | pass |
| TS0012 | Test empty truck | (0,0) (0,0) | 0% | 0% | pass |
| TS0013 | Test full load truck | (2500,100)  (100,10) | -1 | -1 | pass |
| TS0014 | Test over capacity | (2300,85)  (300,20) | -1 | -1 | pass |

**Bugs Found**:

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

**Test Name or ID**: distance

**Test Type**: Black box

**Description**: Function `distance` calculates the Euclidean distance between two points in a 2D space. This test ensures that the function correctly calculates the distance for various input scenarios. We have positive, negative, mixed coordinates test cases, etc.

**Setup:** Visual Studio native unit test

**Test Function**: distance - calculate the Euclidian distance between two points

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0015 | Test same point | {0, 0} {0, 0} | 0 | 0 | pass |
| TS0016 | Test adjacent points | {0, 0} {0, 1} | 1 | 1 | pass |
| TS0017 | Test diagonal | {0, 0} {1, 1} | 1.414214 | 1.414214 | pass |
| TS0018 | Test large distance | {0, 0} {3, 4} | 5 | 5 | pass |
| TS0019 | Test negative coordinates | {-1, -1} {-4, -5} | 5 | 5 | pass |
| TS0020 | Test mixed coordinates | {-1, -1} {1, 1} | 2.828427 | 2.828427 | pass |
| TS0021 | Test positive coordinates | {2, 3} {5, 7} | 5 | 5 | pass |
| TS0022 | Test non integer result | {1, 2} {4, 6} | 5 | 5 | pass |
| TS0023 | Test fractional coordinates | {0, 0} {1, 2} | 2.236068 | 2.236068 | pass |
| TS0024 | Test zero row | {0, 5} {0, -5} | 10 | 10 | pass |
| TS0025 | Test zero column | {7, 0} {-7, 0} | 14 | 14 | pass |

**Bugs Found**:

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

**Test Name or ID**: getClosestPoint

**Test Type**: Black box

**Description**: Function `getClosestPoint` calculates the Euclidean distance from every point on a given route to a single reference point and returns the index of the point in the route that is closest to the reference point. The purpose is to ensure that the function accurately identifies the closest point in various scenarios. We have different points, different routes, and edge test cases.

**Setup:** Visual Studio native unit test

**Test Function**: getClosestPoint - calculates the Euclidian distance from every point in a route to a single point and returns the index of the point in the route which is closest to the point.

**Test Scenarios:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0026 | Test empty route | { 0 }  { 0, 0} | -1 | -1 | pass |
| TS0027 | Test single point | { {{0, 0}}, 1, 'A' }  { 1, 1 } | 0 | 0 | pass |
| TS0028 | Test multiple points | { {{0, 0}, {1, 1}, {2, 2}}, 3, 'A' }  { 1, 0 } | 0 | 0 | pass |
| TS0029 | Test multiple points with closest in the middle of route | { {{0, 0}, {1, 1}, {2, 2}}, 3, 'A' }  { 1, 1 } | 1 | 1 | pass |
| TS0030 | Test multiple points with closest at the last route | { {{0, 0}, {1, 1}, {2, 2}}, 3, 'A' }  { 3, 3 } | 2 | 2 | pass |
| TS0031 | Test long route | {{0, 0}, {1, 2}, {3, 1}, {4, 4}, {5, 0}, {6, 6}, {7, 3}, {8, 8}, {9, 1}, {10, 10},{11, 11}, {12, 2}, {13, 13}, {14, 14}, {15, 5}, {16, 16}, {17, 7}, {18, 18}, {19, 9}, {20, 20}},20, 'B'}  { 10, 9 } | 9 | 9 | pass |
| TS0032 | Test long route with exact match | { { {0, 0}, {1, 2}, {3, 1}, {4, 4}, {5, 0}, {6, 6}, {7, 3}, {8, 8}, {9, 1}, {10, 10},{11, 11}, {12, 2}, {13, 13}, {14, 14}, {15, 5}, {16, 16}, {17, 7}, {18, 18}, {19, 9}, {20, 20} }, 20, 'B'}  { 7, 3 } | 6 | 6 | pass |
| TS0033 | Test long route without match | { {0, 0}, {1, 2}, {3, 1}, {4, 4}, {5, 0}, {6, 6}, {7, 3}, {8, 8}, {9, 1}, {10, 10},{11, 11}, {12, 2}, {13, 13}, {14, 14}, {15, 5}, {16, 16}, {17, 7}, {18, 18}, {19, 9}, {20, 20}},20, 'B'} | 19 | 19 | pass |

**Bugs Found**:

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

**Test Name or ID**: isValidWeight

**Test Type**: Black box

**Description**: we are testing the function “isValidWeight” to ensure that the valid input of a shipment weight. To meet requirement, the shipment weight should above zero and eligible up to 2500kg at the most. If the result is 1, valid, zero is not valid.

**Setup:** Visual Studio native unit test framework

**Test Function**: isValidWeight - check if the weight of shipment is within valid value.

**Test Scenarios: see the chart below**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0034 | Test valid weight,  below limit | {0, 1 {0,0}} | 0/False | 0 | Pass |
| TS0035 | Test valid weight,  Above limit | {2501, 1 {0,0}} | 0/False | 0 | Pass |
| TS0036 | Test valid weight,  Lower limit | {1,1 {0,0}} | 1/True | 1 | pass |
| TS0037 | Test valid weight,  Upper limit | {2500,1 {0,0}} | 1/True | 1 | pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Bugs Found**:

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

**Test Name or ID**: IsValidSize

**Test Type**: Black box

**Description**: Testing shipment size to ensure if the shipment is validated. It should be szie 1 or 3 or 5.

f the size of shipment equal to 1 , 3, 5, it returns 1, which is valid. If zero returned, it is invalid.

**Setup:** Visual Studio native unit test

**est Function**: IsValidSize to see if the size of shipment is within the validate size

**Test Scenarios: see the chart below**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0038 | Test with valid sizes of shipment | {1,1 {0,0}} | 1/True | 1 | Pass |
| TS0039 | Test with valid size | {1, 3 {0,0}} | 1/True | 1 | Pass |
| TS0040 | Test with valid size | {1,5 {0,0}} | 1/Ture | 1 | Pass |
| TS0041 | Test with invalid size | { 1, 0 { 0,0}} | 0/False | 0 | Pass |
| TS0042 | Test with invalid size | { 1, 2 { 0,0}} | 0/False | 0 | Pass |
| TS0043 | Test with invalid size | { 1, 4 { 0,0}} | 0/False | 0 | Pass |
| TS0044 | Test with invalid size | { 1, 6 { 0,0}} | 0/False | 0 | Pass |

**Bugs Found**:

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

**Test Name or ID**: isTruckCanShip

**Test Type**: Black box

**Description**: validate the truck can ship the shipment. The size of shipment should be validated, and the net weight of shipments should be validated as well to pass this test. It is to see the truck’s current shipment weight is still under capacity.

**Setup: Visual Studio unit test framework**

**Test Function**: int isTruckCanShip(struct Truck \*truck, struct Shipment \*shipment); - To see if the current truck is eligible to handle shipment.

**Test Scenarios: below chart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0044 | Test truck with valid weight and valid size | {2300,1{ 0,0}} | 1 True | 1 | Pass |
| TS0045 | Test truck with valid weight and invalid size of shipment | {2301,2{ 0,0}} | 0 Fail | 0 | Pass |
| TS0046 | Test truck with valid weight and invalid size of shipment | {2299,4{ 0,0}} | 0 Fail | 0 | Pass |
| TS0047 | Test truck with invalid weight and invalid size of shipment | {0,6 { 0,0}} | 0 Fail | 0 | Pass |
| TS0048 | Truck with valid weight and valid size | {1, 1, {0,0)) | 1 True | 1 | Pass |
| TS0049 | Truck with valid weight and invalid size | {1, 2, {0,0)) | 0 False | 0 | Pass |
| TS0050 | Truck with valid weight and valid size | {1, 3, {0,0)) | 1 True | 1 | Pass |
| TS0051 | Truck with valid weight and invalid size | {1, 4, {0,0)) | 0 False | 0 | Pass |
| TS0052 | Truck with valid weight and valid size | {1, 5, {0,0)) | 1 True | 1 | Pass |
| TS0053 | Truck with valid weight and invalid size | {1, 6, {0,0)) | 0 False | 0 | Pass |
|  |  |  |  |  |  |

**Bugs Found**:

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

**Test Name or ID**: TestFindTruckForShipment\_whitebox 1 –4

**Test Type**: whitebox

**Description**: This function search available truck after verifying whether the truch has more vomlum and weight available than the shipment’s and whether the destination of shipment is on the truck’s route. Once one of trucks is available, the function return the available truck’s number. If it returns –1, it means no truck is available. In the test data, truck[i] represents that the function checks all the trucks to find eligible.

**Setup:** Visual Studio native unit test

**Test Function**: FindTruckForShipment

**Test Scenarios: see the chart below (max weight is 2500kg max volume is 100 cubic meters)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| TS0054 (whitebox1) | Test with one truck/invalid situationo | (-1,(3,1),truck[0],1(1500,95),-1) | -1 / No truck available | 100% | pass |
| TS0055  (whitebox2) | Test with 3 trucks | (1,(17,1),truck[1],3,(1500,3),1) | 1 / second truck available | 100% | pass |
| TS0056  (whitebox3) | Test with 3 trucks | (-1, (17,11),truck[i],3,(800,),-1) | -1 / No truck available | 100% | pass |
| TS0057  (whitebox4) | Test with 3 trucks | (-1, (15,10),trucks[i],3,(900,5),-1) | -1 / No truck available | 100% | pass |

**Bugs Found**:

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