# TEST DESCRIPTION

**TEST NAME OR ID**: T1-T4

**Test Type**: Black box

**Description**: Test the assignment of appropriate truck with different scenarios.

**Setup:** VS code unit test template, functions and a main() to execute the test function.

**Test Function**: assignShipmentToTruck

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Ideal case in which shipment perfectly fits into the best truck | deliveryMap = { {{0}}, 25, 25 };  fleet = { {1, 1000.0, 10.0, 0, 0, 'B', NULL}, {2, 1500.0, 15.0, 0, 0, 'Y', NULL}, {3, 1200.0, 12.0, 0, 0, 'G', NULL} }  shipmentDetails = {800.0, 8.0, {10, 10}};  totalTrucks = 3; | 1 | 1 | PASS |
| Case in which no suitable truck is available | deliveryMap = { {{0}}, 25, 25 };  struct Truck fleet[3] = { {1, 500.0, 5.0, 0, 0, 'B', NULL}, {2, 600.0, 6.0, 0, 0, 'Y', NULL}, {3, 700.0, 7.0, 0, 0, 'G', NULL} };  shipmentDetails = {800.0, 8.0, {10, 10}};  totalTrucks = 3; | -1 | -1 | PASS |
| Case in which multiple suitable trucks are available | deliveryMap = { {{0}}, 25, 25 };  fleet= { {1, 1000.0, 10.0, 0, 0, 'B', NULL}, {2, 1500.0, 15.0, 0, 0, 'Y', NULL}, {3, 2000.0, 20.0, 0, 0, 'Y', NULL} };  shipmentDetails = {1000.0, 12.0, {10, 10}};  totalTrucks = 3; | 1 | 1 | PASS |
| Case in which shipment exceeds all truck capacities | deliveryMap = { {{0}}, 25, 25 };  fleet[3] = { {1, 500.0, 5.0, 0, 0, 'B', NULL}, {2, 600.0, 6.0, 0, 0, 'Y', NULL}, {3, 700.0, 7.0, 0, 0, 'G', NULL} };  shipmentDetails = {800.0, 8.0, {10, 10}};  totalTrucks = 3; | -1 | -1 | PASS |

**TEST NAME OR ID**: T5-T8

**Test Type**: Black box

**Description**: Test the validation of shipment details entered by the user for different scenarios.

**Setup:**

**Test Function**: validateShipment

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Ideal case in which shipment has valid size and weight | shipment = {500.0, 5.0, {10, 10}}; | 1 | 1 | PASS |
| Case in which shipment has minimum valid size and weight | shipment = {1.0, 1.0, {10, 10}}; | 1 | 1 | PASS |
| Case in which the size and weight is 0 | {0.0, 0.0, {10, 10}}; | 0 | 0 | PASS |
| Case in which the package exceeds size limit | shipment = {500.0, 120.0, {10, 10}}; | 0 | 0 | PASS |

**TEST NAME OR ID**: T9-T12

**Test Type**: Black box

**Description**: Test the accuracy of the display of route that the truck used to deliver the package to its destination.

**Setup:** VS code unit test template, functions and a main() to execute the test function.

**Test Function**: displayRouteToDestination

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Case in which route includes the destination point | struct Route route = { {{0, 0}, {0, 1}, {0, 2}, {1, 2}, {2, 2}}, 5 };  struct Point destination = { 2, 2 }; | 1 | 1 | PASS |
| Case in which destination is not on route | struct Route route = { {{0, 0}, {0, 1}, {0, 2}, {1, 2}, {2, 2}}, 5 };  struct Point destination = { 3, 3 }; | 1 | 1 | PASS |
| Case in which route has only one point | struct Route route = { {{5, 5}}, 1 };  struct Point destination = { 5, 5 }; | 1 | 1 | PASS |
| Case in which route contains maximum number of points | struct Route route = { {{0, 0}, {0, 1}, ..., {24, 24}}, 100 };  struct Point destination = { 24, 24 }; | 1 | 1 | PASS |

**TEST NAME OR ID**: T13-T16

**Test Type**: Black box

**Description**: Test if the function accurately checks if truck has sufficient capacity to hold the shipment.

**Setup:** VS code unit test template, functions and a main() to execute the test function.

**Test Function**: isTruckCapacitySufficient

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Ideal case | truck = {123, 2000.0, 80.0,10, 0, ‘B’,{NULL} }  shipment = {500.0, 10.0, {10,5} } | 1 | 1 | PASS |
| Case in which shipment exactly fits remaining capacity | truck = {123, 500.0, 10.0,10, 0, ‘B’,{NULL} }  shipment = {500.0, 10.0, {10,5} } | 1 | 1 | PASS |
| Case in which shipment exceeds weight capacity | truck = {123, 400.0, 50.0,10, 0, ‘B’,{NULL} }  shipment = {450.0, 5.0, {10,5} } | 0 | 0 | PASS |
| Case in which shipment exceeds volume capacity | truck = {123, 1000.0, 5.0,10, 0, ‘B’,{NULL} }  shipment = {200.0, 6.0, {10,5} } | 0 | 0 | PASS |

**TEST NAME OR ID**: T017-T20

**Test Type**: Black box

**Description**: Test if the function updates the truck's available weight and volume after adding the shipment correctly.

**Setup:** VS code unit test template, functions and a main() to execute the test function.

**Test Function**: updateTruckCapacity

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Standard Case | struct Truck truck = {1, 2500.0, 100.0, 0, 0, 'A', ""};  struct PackageInfo shipment = {500.0, 20}; | 1 | 1 | PASS |
| Shipment size and weight and truck’s remaining size and weight is same | struct Truck truck = {2, 500.0, 100.0, 0, 0, 'B', ""};  struct PackageInfo shipment = {500.0, 20}; | 1 | 1 | PASS |
| Shipment exceeds weight capacity | struct Truck truck = {3, 2500.0, 20.0, 0, 0, 'C', ""};  struct PackageInfo shipment = {500.0, 20.0}; | 1 | 1 | PASS |
| |  | | --- | | **Case in which truck is NULL** | | |  | | --- | | Case in which truck is NULL | | 0 | 0 | PASS |

**TEST NAME OR ID**: T21-T24

**Test Type**: Black box

**Description**: Test if the function finds the nearest available truck to the shipment's pickup location accurately.

**Setup:** VS code unit test template, functions and a main() to execute the test function.

**Test Function**: findNearestTruck

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Standard Case | map = { {{'.'}}, 10, 10 fleet= [{101, {2, 3}}, {102, {5, 5}}, { 103, {1, 1}}]  totalTrucks= 3  pickupLocation= {2, 2} | 0 | 0 | PASS |
| Case in which multiple trucks are available at same distance | map = { {{'.'}}, 10, 10 fleet= [{101, {2, 2}}, {102, {2,2}}, { 103, {3,3}}]  totalTrucks= 3  pickupLocation= {2, 2} | 101  This is because the first truck will be picked up if multiple trucks are available. | 101 | PASS |
| Case in which no trucks are available | map = { {{'.'}}, 10, 10 fleet= []  totalTrucks= 0  pickupLocation= {2, 2} | -1 | -1 | PASS |
| Case in which truck is at the minimum possible distance | map = { {{'.'}}, 100, 100 fleet= [{101, {20,0}}, {102, {99,99}}, { 103, {50,50}}]  totalTrucks= 3  pickupLocation= {0,0} | 0 | 0 | PASS |

**TEST NAME OR ID**: T25-28

**Test Type**: Black box

**Description**: Test if the function can initialize the truck with distinct parameters

**Setup:** VS code unit test template, functions, and a main() to execute the test function.

**Test Function**: initializeTruck()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Standard case with valid truck pointer and parameters | struct Truck truck;  int truckID = 1;  char route = 'A'; | 0 | 0 | PASS |
| |  | | --- | | **Case where truck pointer is NULL** |  |  | | --- | |  |  |  | | --- | |  | | struct Truck\* truck = NULL;  int truckID = 1;  char route = 'B'; | 0 | 0 | PASS |
| Case with large truck ID | struct Truck truck;  int truckID = 2147483647;  char route = 'C'; | 0 | 0 | PASS |
| Case with non-standard route character | struct Truck truck;  int truckID = 5;  char route = '#'; | 0 | 0 | PASS |

**TEST NAME OR ID**: T29-T32

**Test Type**: Black box

**Description**: Test if the function is able to validate the inputs for package and initialize it.

**Setup:** VS code unit test template, functions, and a main() to execute the test function.

**Test Function**: initializePackage()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Standard Case | (&package, 500.0f, 3.0f, 10, 5) | 1 | 1 | PASS |
| Case in which weight is negative | (&package, -100.0f, 3.0f, 10, 5) | 0 | 0 | PASS |
| Case in which position of box is out of scope | (&package, 500.0f, 3.0f, 26, 5) | 0 | 0 | PASS |
| Case in which box is at the edge of the map. | (&package, 500.0f, 3.0f, 25, 25) | 1 | 1 | PASS |