# Test Description

**Test Name or ID**: AT001

**Test Type**: Acceptance test

**Description**: Assign the package to the most suitable truck among the given trucks. Suitability is evaluated based on the truck's route, remaining load capacity, and weight and volume limitations.

**Setup:** Visual Studio IDE

**Test Function**: hasEnoughSpace, loadShipment, checkMoreCapacity

**Requirement: R001 -** Assign packages to the optimal truck.

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Verifies that the truck's capacity is sufficient. | Truck capacity (1000 kg, 10 m³), package weight (300 kg), package volume (3 m³). | The hasEnoughSpace function should return 1 indicating sufficient space. | 1 | Pass |
| Verifies that the system correctly handles situations where the truck's load capacity is insufficient. | Truck capacity (200 kg, 2 m³), package weight (300 kg), package volume (3 m³) | 0 (The loadShipment function should output "Truck is full!") | 0 | Pass |
| Checks whether the truck with the greater load capacity is correctly chosen between two options. | Truck 1 (500 kg, 5 m³), Truck 2 (700 kg, 7 m³) | The checkMoreCapacity function should return 1, indicating that Truck 1 is chosen. | 1 | Pass |

**Bugs Found**:

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

# Test Description

**Test Name or ID**: AT002

**Test Type**: Acceptance test

**Description**: Calculate the Euclidean distance between two points (x1, y1) and (x2, y2).

**Setup:** Visual studio IDE.

**Test Function**: distance

Requirement: R002 - Calculate the Euclidean distance between two points.  
**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the distance function to ensure it accurately calculates the Euclidean distance between two given points using the mathematical formula. | Input p1 = 3, 4 p2 = 7, 1 | The distance function should use the formula √((x2 - x1)² + (y2 - y1)²) and return the distance between both points | The distance function should use the formula √((x2 - x1)² + (y2 - y1)²) and return the distance between both points | Pass |

**Bugs Found**:

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

# Test Description

**Test Name or ID**: AT003

**Test Type**: Acceptance Test

**Description**: After assigning the package to the truck, determine the optimal route to the destination.

**Setup:** Visual Studio IDE

**Test Function**: addRoute, loadShipment

**Requirement**: R003 - Assign a package to a truck and determine the route.

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Verifies that the loadShipment function can correctly validate the details of a box and mark the shipment process as successful. | Call loadShipment function and input box details  struct Truck truck = { 1000, 10 };  struct Shipment shipment = { 300, 3, {0, 0} }; | The function should validate the shipment and return 1 if successful | 1 | Pass |
| Tests the addRoute function to assess its capability to analyze provided map data and efficiently determine a travel path to the destination. | Call addRoutefunction and input map data and starting point  struct Point start = { 0, 0 };  destination = { 24, 24 };  struct Route route;  route.numPoints = 0;  route.routeSymbol = 'P'; | The function should analyze the map and plot a destination. | The function analyze the map and plot a destination. | Pass |

**Bugs Found**:

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

# Test Description

**Test Name or ID**: AT004

**Test Type**: Acceptance Test

**Description**: Plan the shortest route from the truck's starting point to the destination.

**Setup:** Visual Studio IDE

**Test Function**: shortestPath

**Requirement**: R004 - Plan the optimal route for the truck.  
**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the shortestPath function to ensure it accurately determines the optimal route by considering the shortest path that avoids obstacles on the map. | **Map Data:**  Grid Size: 25x25  Obstacles: Rows 12, Columns 5-15 marked as obstacles (1)  **Start Point:** (0, 0) **Destination Point:** (24, 24) | The function should analyze the map and determine a route from (0, 0) to (24, 24) that avoids the obstacles and is the shortest possible path. | - | Fail |

**Bugs Found**:

**Bugs**: Infinite loop occurs when calling shortestPath.

**Details**: The function enters an infinite loop due to a logic flaw in the shortestPath function. It fails to terminate properly when no further valid moves are available or when it continuously returns to previously visited points without making progress towards the destination.

**How to Reproduce**:

Call the shortestPath function with the map data where the grid size is 25x25 and rows 12, columns 5-15 are marked as obstacles.

Set the start point at (0, 0) and the destination point at (24, 24).

The function does not exit as expected and causes the program to hang indefinitely.

# Test Description

**Test Name or ID**: AT005

**Test Type**: Acceptance Test

**Description**: Validate that the input package's weight and size are valid.

**Setup:** Visual Studio IDE

**Test Function**: validateBox, validateWeight

**Requirement**: R005 - Validate the weight and size of the package.  
**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the validateWeight and validateBox functions to ensure they can accurately identify and reject invalid package specifications. | Input package weight (-1 kg), package volume (-4m) | validateWeight should return 0 and validateBox should also return 0. | 0, 0 | Pass |

**Bugs Found**:

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

# Test Description

**Test Name or ID**: AT006

**Test Type**: Acceptance Test

**Description**: The maximum load capacity of the truck is 2500kg and the maximum volume is 100m³

Display the current loading status as a percentage.

If there is no remaining capacity, print 'The truck is full'.

**Setup:** Visual Studio IDE

**Test Function**: checkMoreCapacity

**Requirement**: R006 - Check the loading status of the truck.

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the checkMoreCapacity function to determine which truck has more available capacity and to provide a status update on the current load as a percentage. | Truck 1 (2000 kg, 95 m³), Truck 2 (1800 kg, 93 m³) | The checkMoreCapacity function should return 2, indicating that Truck 2 is chosen. The function will also return the percentage of space remaining | 2 | Pass |

**Bugs Found**:

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

# Test Description

**Test Name or ID**: AT007

**Test Type**: Acceptance Test

**Description**: Ensure the truck avoids buildings when rerouting.

**Setup:** Visual Studio IDE

**Test Function**: getPossibleMoves

**Requirement**: R007 - Avoid buildings when rerouting.  
**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the getPossibleMoves function to ensure it calculates possible movement paths that avoid any obstacles present on the map. | Call getPossibleMoves function and input map data, starting point, and return path  Map data: Buildings at locations (5,5), (5,6), (5,7) Starting point: (4,4  Return path: (3,4) | The function should return all possible movement paths from the starting point (4,4) avoiding buildings. Expected paths could include (4,3), (4,5), (3,5), (6,4), which are all free of buildings. | - | Fail |

**Bugs Found**:

**Bugs**: Infinite loop occurs when calling shortestPath.

**Details**: This indicates a critical issue where the shortestPath function fails to terminate under certain conditions, possibly due to not properly updating its state or encountering a scenario where it continually revisits the same points without progress. Further investigation is needed to identify the specific conditions triggering the loop, such as specific start and end points or configurations of obstacles on the map.

**How to Reproduce**:

Call getPossibleMoves function and input map data, starting point, and return path

Map data: Buildings at locations (5,5), (5,6), (5,7) Starting point: (4,4)

Return path: (3,4)The function does not exit as expected and causes the program to hang indefinitely.

# Test Description

**Test Name or ID**: AT008

**Test Type**: Acceptance Test

**Description**: Visualize the routes taken by the trucks to facilitate understanding of route planning.

**Setup:** Visual Studio IDE

**Test Function**: getBlueRoute, getGreenRoute, getYellowRoute

**Requirement**: R008 - Visualize Truck Routes

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the getBlueRoute, getGreenRoute, and getYellowRoute functions to ensure they correctly visualize the truck routes on a map. | 1. populateMap() is called to initialize the base map.  2. getBlueRoute(), getGreenRoute(), getYellowRoute() are each called to obtain truck route data.  3. addRoute() is used to add each route to the map. | A map filled in with colored routes which also highlights positions and destinations of the trucks | The routes is not correct. | Fail |

**Bugs Found**:

Incorrect Route Data

Description: If getBlueRoute functions return incorrect route data, the routes may not be plotted correctly on the map, leading to failed assertions.

How to Reproduce: 1. populateMap() is called to initialize the base map.

2. getBlueRoute(), getGreenRoute(), getYellowRoute() are each called to obtain truck route data.

3. addRoute() is used to add each route to the map.

# Test Description

**Test Name or ID**: AT009

**Test Type**: Acceptance Test

**Description**: Record whether a package has been successfully delivered.

**Setup:** Visual Studio IDE

**Test Function**: eqPt

**Requirement**: R009 - Record Package Delivery Status

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Tests the eqPt function to ensure it correctly determines when a truck has reached its delivery destination. | Call eqPt function and input current pos (6, 6)and dest pos (6, 6) | The function should return 1 indicating that the truck has reached its destination | The function should return 1 indicating that the truck has reached its destination | Pass |

**Bugs Found**:

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