# Test Description

**Test Name or ID**: 01

**Test Type**: White Box

**Description**: Validate the correctness of the `isPackageValid` function by testing different paths for weight and volume conditions.

**Setup:** Define constants for `SMALL\_BOX`, `MEDIUM\_BOX`, and `LARGE\_BOX`. Ensure the `isPackageValid` function is accessible

**Test Function**: `isPackageValid(double weight, double volume)`

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Valid weight and volume | 10.0, SMALL\_BOX | 1 (Valid) | 1 (Valid) | Pass |
| Invalid weight (less than 1) | 0.5, MEDIUM\_BOX | 0 (Invalid) | 0 (Invalid) | Pass |
| Invalid weight (greater than 1000) | 1500.0, LARGE\_BOX | 0 (Invalid) | 0 (Invalid) | Pass |
| Valid weight but invalid volume | 500.0, 2.0 | 0 (Invalid) | 0 (Invalid) | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No bugs were found during testing. All outputs matched the expected results.

**Test Name or ID**: 02

**Test Type**: White Box

**Description**: Validate the correctness of the `isDestValid` function by testing different paths for destination validity in a map grid.

**Setup:** Prepare a map structure with predefined rows, columns, and grid values. Ensure the `isDestValid` function is accessible.

**Test Function**: `isDestValid(const struct Map\* map, const struct Point dest)`

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Valid destination within bounds and not blocked | Map: 5x5, Dest: (2,2) | 1 (Valid) | 1 (Valid) | Pass |
| Destination out of bounds (negative indices) | Map: 5x5, Dest: (-1,2) | 0 (Invalid) | 0 (Invalid) | Pass |
| Destination out of bounds (exceeding indices) | Map: 5x5, Dest: (5,2) | 0 (Invalid) | 0 (Invalid) | Pass |
| Destination blocked (value = 1) | Map: 5x5, Dest: (3,3) [Blocked] | 0 (Invalid) | 0 (Invalid) | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No bugs were found during testing. All outputs matched the expected results.

**Test Name or ID**: 03

**Test Type**: White Box

**Description**: Validate the behavior of the `addPackage` function when adding a package to a truck. It tests scenarios such as adding to an empty truck, a partially full truck, or when exceeding the truck’s capacity

**Setup:** Prepare a truck structure and package data. Ensure the truck is initialized correctly, and the `addPackage` function is accessible.

**Test Function**: `addPackage(struct Truck\* truck, const struct Package\* package)`

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Add a package to an empty truck | Truck: Empty, Package: {Weight: 50kg, Volume: 1m³} | Truck contains 1 package, updated weight and volume | Truck contains 1 package, updated weight and volume | Pass |
| Add a package to a partially full truck | Truck: {Weight: 500kg, Volume: 10m³}, Package: {Weight: 50kg, Volume: 1m³} | Truck contains the package, updated weight and volume | Truck contains the package, updated weight and volume | Pass |
| Exceed truck’s weight capacity | Truck: {Weight: MAX\_WEIGHT - 10kg}, Package: {Weight: 20kg} | Truck rejects the package | Truck rejects the package | Pass |
| Exceed truck’s volume capacity | Truck: {Volume: MAX\_VOLUME - 1m³}, Package: {Volume: 2m³} | Truck rejects the package | Truck rejects the package | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No bugs were found during testing. All outputs matched the expected results.

**Test Name or ID**: 04

**Test Type**: White Box

**Description**: Validate the behavior of the `shortestPath` function when calculating the shortest path between a start and destination point on a map. It tests scenarios such as valid paths, paths blocked by obstacles, unreachable destinations, and edge cases.

**Setup:** Prepare a map structure with predefined rows, columns, and grid values. Ensure start and destination points are initialized correctly, and the `shortestPath` function is accessible.

**Test Function**: `shortestPath(const struct Map\* map, const struct Point start, const struct Point dest)`

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Calculate shortest path on an obstacle-free map | Map: 5x5 grid, Start: (0,0), Dest: (4,4) | Valid route found with minimum distance | Valid route found with minimum distance | Pass |
| Destination unreachable due to obstacles | Map: 5x5 grid with all paths blocked, Start: (0,0), Dest: (4,4) | No valid route, return empty route | No valid route, return empty route | Pass |
| Start and destination are the same | Map: 5x5 grid, Start: (2,2), Dest: (2,2) | Route with zero distance | Route with zero distance | Pass |
| Destination out of bounds | Map: 5x5 grid, Start: (0,0), Dest: (5,5) | No valid route, return empty route | No valid route, return empty route | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No bugs were found during testing. All outputs matched the expected results.

**Test Name or ID**: 05

**Test Type**: White Box

**Description**: Validate the behavior of the calcDeliveryPath function when determining the closest route point to a given destination and calculating the route distance.

**Setup:** Prepare a map structure and a route with multiple points. Ensure the calcDeliveryPath function is accessible.

**Test Function**: calcDeliveryPath(const struct Map\* map, const struct Route\* route, const struct Point\* point)

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Find closest point to destination | Map: 5x5, Route: [(0,0), (2,2), (4,4)], Point: (3,3) | Closest point: (2,2), Distance calculated | Closest point: (2,2), Distance calculated | Pass |
| Destination matches a route point | Map: 5x5, Route: [(0,0), (2,2), (4,4)], Point: (2,2) | Closest point: (2,2), Distance: 0 | Closest point: (2,2), Distance: 0 | Pass |
| No valid points in route | Map: 5x5, Route: [], Point: (2,2) | No route found, Distance: MAX\_DIST | No route found, Distance: MAX\_DIST | Pass |
| Multiple equidistant points | Map: 5x5, Route: [(0,0), (2,2), (4,4)], Point: (3,3) | Closest point selected based on implementation | Closest point selected based on implementation | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No bugs were found during testing. All outputs matched the expected results.

**Test Name or ID**: 06

**Test Type**: White Box

**Description**: Validate the behavior of the isPackageAcceptable function when checking if a package can be added to a truck without exceeding weight or volume capacity.

**Setup:** Prepare a truck structure with predefined current weight and volume values. Ensure the isPackageAcceptable function is accessible.

**Test Function**: isPackageAcceptable(const struct Truck\* truck, const struct Package\* package)

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Package fits within capacity | Truck: {Weight: 500kg, Volume: 10m³}, Package: {Weight: 50kg, Volume: 2m³} | 1 (Acceptable) | 1 (Acceptable) | Pass |
| Exceeds weight capacity | Truck: {Weight: MAX\_WEIGHT - 10kg}, Package: {Weight: 20kg, Volume: 2m³} | 0 (Not Acceptable) | 0 (Not Acceptable) | Pass |
| Exceeds volume capacity | Truck: {Weight: 500kg, Volume: MAX\_VOLUME - 1m³}, Package: {Weight: 50kg, Volume: 2m³} | 0 (Not Acceptable) | 0 (Not Acceptable) | Pass |
| Exceeds both weight and volume | Truck: {Weight: MAX\_WEIGHT - 5kg, Volume: MAX\_VOLUME - 0.5m³}, Package: {Weight: 10kg, Volume: 1m³} | 0 (Not Acceptable) | 0 (Not Acceptable) | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No bugs were found during testing. All outputs matched the expected results.