# Test Description

**Test Name or ID**: TB01

**Test Type**: Black Box

**Description**: Checks if the user input for the weight of a shipment is within a valid range, then returns 1 and is invalid if this returns 0

**Setup:** Under the visual studio 2022 environment.

**Test Function**: validWeight()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Nominal input | Weight = 1 | 1 | 0 | Fail |
| Nominal input | Weight = 500 | 1 | 0 | Fail |
| Nominal input | Weight = 1000 | 1 | 0 | Fail |
| Exit test | Weight = 0 | 1 | 0 | Fail |
| Value larger than Max | Weight = 1000.5 | 0 | 0 | Pass |
| Value smaller than Min | Weight = -1 | 0 | 0 | Pass |
|  |  |  |  |  |

**Bugs Found**:

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

**Test Name or ID**: TB02

**Test Type**: Black Box

**Description**: Checks if the user input for the size of a shipment is within a valid range (0, 0.25, 0.5 and 1), then returns 1 and is invalid if this returns 0

**Setup:** Under the visual studio 2022 environment.

**Test Function**: validSize()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Nominal input | size = 0.25 | 1 | 1 | Pass |
| Nominal input | size = 0.5 | 1 | 1 | Pass |
| Nominal input | size = 1 | 1 | 1 | Pass |
| Exit test | size = 0 | 1 | 0 | Fail |
| Value larger than Max | size = 2 | 0 | 0 | Pass |
| Value smaller than Min | size = - 0.25 | 0 | 0 | Pass |
| ~~Invalid size option~~ | ~~size = 0.125~~ | ~~0~~ |  |  |
| ~~Invalid size option~~ | ~~size = 0.75~~ | ~~0~~ |  |  |

**Bugs Found**:

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

**Test Name or ID**: TB03

**Test Type**: Black Box

**Description**: Checks if the user input for the destination is valid which should be on a building square and is invalid if it returns 0

**Setup:** Under the visual studio 2022 environment.

**Test Function**: validDestination()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Nominal input | 1, 1 | 1 | Can’t run |  |
| Nominal input | 11, 11 | 1 | Can’t run |  |
| Nominal input | 23, 24 | 1 | Can’t run |  |
| ~~Exit test~~ | ~~des = X~~ | ~~1~~ |  |  |
| Row larger than Max | 25, 1 | 0 | Can’t run |  |
| Row smaller than Min | -2, 1 | 0 | Can’t run |  |
| Col larger than Max | 0, 25 | 0 | Can’t run |  |
| ~~Col smaller than Min~~ | ~~des = 10~~ | ~~0~~ |  |  |
| White space (not building) | 9, 12 | 0 | Can’t run |  |
| White space (not building) | 23, 24 | 0 | Can’t run |  |
| Incomplete input | 0, 11 | 0 | Can’t run |  |
| ~~Incomplete input~~ | ~~Des = L~~ | 0 | Can’t run |  |

**Bugs Found**:

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

**Test Name or ID**: TB004

**Test Type**: Black box

**Description**: This test aims to verify if the “isTruckFull” function correctly determines if a truck can accommodate a shipment based on weight and space constraints.

**Setup:** Prepare a truck object and a shipment object with varying weights and dimensions.

**Test Function**: int isTruckFull(struct Truck truck, struct Shipment box);

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Test with a truck that has sufficient weight and space for the shipment | truck: {1000, 36},  box: {500, 18} | 0 | 1 | Fail |
| Test with a truck that is at maximum weight and space capacity | truck: {1000, 36},  box: {1000, 36} | 0 | 1 | Fail |
| Test with a truck that is already exceeding its weight capacity but has sufficient space | truck: {1000, 36},  box: {2000, 18} | 1 | 0 | Fail |
| Test with a truck that has sufficient weight capacity but lacks space for the shipment | truck: {1000, 36},  box: {500, 50} | 1 | 0 | Fail |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

N/A

**Test Name or ID**: TB005

**Test Type**: Black box

**Description**: The function "calcShortPathDist" calculates the total distance of a given route and needs to be tested for accuracy.

**Setup:** Prepare test data with various route scenarios (e.g., empty route, straight line route, closed loop route) and set up a programming environment with access to the code. Use assertions to compare the actual total distance calculated by the function with the expected results.

**Test Function**: double calcShortPathDist (struct Route shortP);

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Test numPoints with the same points in shortP | shortP = { 0,2 } | 0.0 | 0.0 | Pass |
| Test numPoints less than the points in shortP | shortP = { 1,0 } | 1.0 | 0.0 | Fail |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

N/A

**Test Name or ID**: TB006

**Test Type**: Black box

**Description**: The function "convLetter" takes a character representing a letter (uppercase) and converts it into its corresponding numeric value by ASCII codes.

**Setup:** Prepare the letterCord as an uppercase letter.

**Test Function**: int convLetter(char letterCord);

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| **Test letterCord as A** | **letterCord = 'A';** | 65 | 0 | Fail |
| Test letterCord as Z | **letterCord = 'Z';** | 90 | 25 | Fail |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

N/A

**~~Test Name or ID~~**~~: TB007~~

**~~Test Type~~**~~: Black box~~

**~~Description~~**~~: This test aims to verify if the "findIndexOfShortestDist" function correctly determines the index of the shortest calculated distance in a truck route.~~

**~~Setup:~~** ~~Prepare a distance list array for a truck route.~~

**~~Test Function~~**~~: int findIndexOfShortestDist(double distanceList[]);~~

**~~Test Scenarios:~~**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ~~Description~~ | ~~Test Data~~ | ~~Expected Result~~ | ~~Actual Result~~ | ~~Pass/Fail~~ |
| ~~Test with a distance list array containing positive distances.~~ | ~~distanceList: [1.1, 2.2, 3.3, 4.4, 5.5]~~ | ~~Index of the shortest distance in the distance list array~~ | ~~The shortest distance in the array~~ | ~~Pass~~ |
| ~~Test with a distance list array containing negative distances.~~ | ~~distanceList: [-5.5, -4.4, -3.3, -2.2, -1.1]~~ | ~~Index of the shortest distance in the distance list array~~ | ~~The shortest distance in the array~~ | ~~Pass~~ |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

~~Bugs Found:~~

~~N/A~~

**Test Name or ID**:  TB08

**Test Type**: Black Box

**Description**: Testing the determineDivergencePath function to ensure it behaves as expected.

**Setup:** Initialize a truck structure with a specific route and call the determineDivergencePath function.

**Test Function**: determineDivergencePath

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **Test when the truck's path has no points** | truckList[0].s\_Path.numPoints = 0 | No changes should be made to the truck's path. | Call the function and check if the truck's path has been modified correctly. | Pass |
| **Test when the truck's path has multiple points.** | (1)truckList[0].s\_Path.numPoints = 3  (2)truckList[0].s\_Path.points[0] = Point(5, 5) | The determineDivergencePath function should modify the truck's path based on the divergence logic. | Call the function and check if the truck's path has been modified correctly. | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

None

**Test Name or ID**: TB09

**Test Type**: Black box

**Description**: Testing the setValidDefaultMinTruck function to ensure it returns the correct index of the truck.

**Setup:** Create a truckList array with different truck structures and call the setValidDefaultMinTruck function.

**Test Function**: setValidDefaultMinTruck

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **Test when there is a valid truck with a non-zero number of points.** | * truckList[0].s\_Path.numPoints = 0 * truckList[1].s\_Path.numPoints = 5 * truckList[2].s\_Path.numPoints = 0 | The function should return the index of the truck with non-zero points, which is 1. | Call the function and compare the returned index with the expected result | Pass |
| **Test when all trucks have a non-zero number of points in their paths.** | * truckList[0].s\_Path.numPoints = 3 * truckList[1].s\_Path.numPoints = 4 * truckList[2].s\_Path.numPoints = 2 | The function should return -1 to indicate that no truck can be considered as the default minimum truck. | Call the function and compare the returned value with the expected result. | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

None

**Test Name or ID**: TB10

**Test Type**: Black box

**Description**: Testing the findBestTruckForShipment function to ensure it returns the correct index of the truck for the given shipment.

**Setup:** Create a map, truckList array, and shipment structure with specific values. Call the findBestTruckForShipment function.

**Test Function**: findBestTruckForShipment

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **Test when there is a valid truck with the shortest path and not fully loaded.** | * Map m with specific buildings and paths * truckList array with at least one truck having a non-zero number of points in its path * Shipment package with specific destination and weight * isTruckFull function returning false for the selected truck | The function should return the index of the truck with the shortest path that is not fully loaded. | Call the function and compare the returned index with the expected result. | Pass |
| **Test when all trucks have the same number of points in their paths.** | * Map m with specific buildings and paths * truckList array with three trucks having the same number of points in their paths * Shipment package with specific destination and weight * isTruckFull function returning false for all the trucks | The function should return the index of the truck that has the most room for space and weight for the shipment(not fully loaded). | Call the function and compare the returned index with the expected result. | Pass |
| **Description: Test when no truck has a non-zero number of points in its path. Test Data:** | * Map m with specific buildings and paths * truckList array with all trucks having a numPoints value of 0 * Shipment package with specific destination and weight | The function should return -1 to indicate that no truck can accommodate the shipment. | Call the function and compare the returned value with the expected result. | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

None

**Test Name or ID**: TB11

**Test Type**: Black Box

**Description**: Calculate total percentage of weight and space to decide which truck to load if they are the same distance from the delivery

**Setup:** Pass in a truck structure to calculate and return a combined total percentage of the weight and space

**Test Function**: capPercentage()

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Nominal input test | t.curSpace = 20 t.curWeight = 500 | 1.055 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

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