**Test cases**

**BB01:**

**Test Function**: isValidWeight

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| BB01-A  Checks if the total capacity and weight is less than the maximum values and whether the package is added to the truck | package.capacity=4;  truck.capacityHolding=30;  package.weight=100;  truck.weightHolding=1000; | 1 | 1 | Pass |
| BB01-B  Checks if the package is added when the capacity exceeds maximum limit but total weight is within limits | package.capacity=4;  truck.capacityHolding=47;  package.weight=900;  truck.weightHolding=200; | 0 | 0 | Pass |
| BB01-C  Checks if the package id added when the total capacity is within limits but the total weight exceeds the maximum limit. | package.capacity=4;  truck.capacityHolding=15;  package.weight=1200;  truck.weightHolding=300; | 0 | 0 | Pass |
| BB01-D  Checks if the package can be added onto an empty truck | Package.capacity=4;  Truck.capacityHolding=0;  package.weight=500;  truck.weightHolding=0; | 1 | 1 | Pass |

**Bugs Found**:

No bugs at the moment because we conducted blackbox testing.

**BB02:**

**Test Function**: int promptUser(struct Package\* package)

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| BB02-A  This test is to check weather the function works with valid data. | Weight = 20  Capacity = .5  Destination = 12L | 1 | 1 | Pass |
| BB02-B  Now let’s try to input too much weight for a truck to handle. | Weight = 1505  Capacity = .5  Destination = 12L | 0 | 0 | Pass |
| BB02-C  The destination is invalid and the data does not have to be assigned. | Weight = 20  Capacity = .5  Destination = 28x | 0 | 0 | Pass |
| BB02-D  We are allowed to put ½, 1 or 2 cubic meters boxes, not much or less. | Weight = 20  Capacity = 3  Destination = 12L | 0 | 0 | Pass |
| BB02-E  This input means that the application has to stop. The next step of the main function will be checking if the package structure data is equal to the provided test data, and if it is print thanking message and stop the program | Weight = 0  Capacity = 0  Destination = x | 1 | 1 | Pass |
| BB02-F  If destination is not equal to x, weight and capacity have to be positive numbers to pass and vise versa. | Weight = 0  Capacity = 0  Destination = 12L | 0 | 0 | Pass |
| BB02-G  If destination is not equal to x, weight and capacity have to be positive numbers to pass and vise versa. | Weight = 5  Capacity = ½  Destination = x | 0 | 0 | Pass |

**Bugs Found**:

No bugs found for now.

**BB03:**

**Test Function**: dispatchTruck

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| BB03-A  To check if an invalid Truck pointer can be handled by the function | Trucks = NULL;  Destination = {‘8’,’G’}  Routes = {{ {{‘8’,’F’}},1,’A’ }} | NULL | NULL | Pass |
| BB03-B  To check if an invalid route can be handled by the function | Trucks = { {0,12.0,NULL,NULL } },  Routes = NULL,  Destination = {‘8’,’G’} | NULL | NULL | Pass |
| BB03-C  To check if a empty destination can be handled | Trucks = { {0,12.0,NULL,NULL } },  Routes = {{ {{‘8’,’G’}},1,’A’ }}  Desination = {‘’,’’} | NULL | NULL | Pass |
| BB03-D  To check if everything is passed on correctly, we are able to get accurate results | Trucks = { {0,12.0,NULL,NULL } },  Routes = {{ {{‘8’,’G’}},1,’A’ }}  Destination = {‘8’,’G’} | {0,12.0, { {{‘8’,’G’}},1,’A’ },NULL } }, | {0,12.0, { {{‘8’,’G’}},1,’A’ },NULL } }, | Pass |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Bugs Found**:

No Bugs Found.

**BB04:**

**Test Function**:double limitingFactor(struct truckInfo tk)

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| BB04-A  Passing values where weight is the limiting factor | { tk.weight = 1200.00,  tk.volume = 15.00,  tk.limitFactor = 0.00} | 80.00  (expecting percentage of weight to be returned as limiting factor) | 80.00 | Pass |
| BB04-B  Passing values where volume is the limiting factor | { tk.weight = 500.00,  tk.volume = 25.00,  tk.limitFactor =0.00} | 52.08  (expecting percentage of volume to be returned as limiting factor) | 52.08 | Pass |
| BB04-C  Passing values where weight is the limiting factor and maximum weight is reached | { tk.weight = 1500.00,  tk.volume = 25.00,  tk.limitFactor =0.00} | 100.00  (expecting percentage of weight to be returned as limiting factor and that too 100%) | 100.00 | Pass |
| BB04-D  Passing values where volume is the limiting factor and maximum volume is reached | { tk.weight = 500.00,  tk.volume = 48.00,  tk.limitFactor =0.00} | 100.00  (expecting percentage of weight to be returned as limiting factor and that too 100%) | 100.00 | Pass |
| BB04-E  Passing values which will in same percentage for weight and volume | { tk.weight =750 .00,  tk.volume = 24.00,  tk.limitFactor =0.00} | 50.00  (expecting percentage of any factor that is weight or volume to be returned as limiting factor as the percentage of both is same) | 50.00 | Pass |
| BB04-F  Passing 0 for both weight and volume | { tk.weight =0 .00,  tk.volume = 0.00,  tk.limitFactor =0.00} | 0.00  (expecting percentage of any factor ,that is weight or volume to be returned as limiting factor as the percentage of both is same) | 0.00 | Pass |
| BB04-G  Passing values that ersult in percentage very close to each other | { tk.weight = 751.00,  tk.volume = 24.00,  tk.limitFactor =0.00  } | 50.06  (expecting percentage of weight as limiting factor as by slight difference it is greater than volume percentage) | 50.06 | Pass |
| BB04-H  Passing decimal value very near to maximum weight | { tk.weight = 1497.00,  tk.volume = 24.00,  tk.limitFactor =0.00  } | 99.8  (expecting exact decimal value for limiting factor) | 99.8 | Pass |
| BB04-I  Passing decimal value very near to maximum volume | { tk.weight = 750.00,  tk.volume = 47.7,  tk.limitFactor =0.00  } | 99.37  (expecting exact decimal value for limiting factor) | 99.37 | Pass |
| BB04-J  Passing values for both weight and volume 1 less than maximum values they can be of | { tk.weight = 1499.00,  tk.volume = 47.00,  tk.limitFactor =0.00  } | 99.94  (expecting weight to be limiting factor because although both the values are one less than maximum but percentage for weight is more) | 99.94 | Pass |

**Bugs Found**:

No bugs were found as of now based on black box test cases.

**BB05:**

**Test Function**: distance

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Adjacent Points on the Map  Test Name or ID: BB05-A  Description: Checks distance calculation for two adjacent points on the map. | p1 = {row: 1, col: 1} (Equivalent to 1A)  p2 = {row: 1, col: 2} (Adjacent to 1A on the right, equivalent to 1B) | Distance = 1 (unit distance on the grid) | 1 | Pass |
| Diagonal Points on the Map  Test Name or ID: BB05-B  Description: Calculates distance for two points diagonally spaced on the map. | p1 = {row: 1, col: 1} (Equivalent to 1A)  p2 = {row: 2, col: 2} (Diagonal to 1A, equivalent to 2B) | Distance = sqrt(2) ≈ 1.414 | 1.414 | Pass |
| Points on the Same Row  Test Name or ID: BB05-C  Description: Verifies distance calculation for two points on the same row. | p1 = {row: 5, col: 8} (Equivalent to 5H)  p2 = {row: 5, col: 20} (Equivalent to 5T) | Distance = 12 (units along the same row) | 12 | Pass |
| Points on the Same Column  Test Name or ID: BB05-D  Description: Tests the function with two points on the same column. | p1 = {row: 10, col: 15} (Equivalent to 100)  p2 = {row: 20, col: 15} (Equivalent to 200) | Distance = 10 (units along the same column) | 10 | Pass |

**Bugs Found**:

No bugs at the moment because we conducted blackbox testing.

**BB06:**

**Test Function**: shortestPath

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Starting at the Office to a Close Destination  Test Name or ID: BB06 -A  Description: Calculates the shortest path from the office to a nearby destination without any route diversions. | Start Point (Office): 1A  Destination Point: 5D | Direct path to 5D, avoiding buildings. | Direct path to 5D, avoiding buildings. | Pass |
| Starting at the Office to a Destination on Blue Route  Test Name or ID: BB06-B  Description: Calculates the shortest path from the office to a destination along the blue route. | Start Point (Office): 1A  Destination Point: 12J | Path follows the blue route up to 12J. | Path follows the blue route up to 12J. | Pass |
| Cross-Route Delivery  Test Name or ID: BB06-C  Description: Checks the shortest path calculation from a point on the blue route to a point on the green route. | Start Point: 12J  Destination Point: 17Q | Path navigates from blue to green route, avoiding buildings. | Path navigates from blue to green route, avoiding buildings. | Pass |
| Long Distance Delivery  Test Name or ID: BB06-D  Description: Assesses the function's capability for a long-distance delivery from one corner of the map to the opposite corner. | Start Point (Office): 1A  Destination Point: 25Y | Efficient path using routes as long as possible and avoiding buildings. | Efficient path using routes as long as possible and avoiding buildings. | Pass |

**Bugs Found**:

No bugs at the moment because we conducted black box testing.