Testing Report

Blackbox Testing Report

findTruckShipment

Purpose: This function identifies the most suitable truck to handle a given shipment based on predefined routes and truck capacities.

The findTruckShipment function was tested across five scenarios:

- **T001:** No suitable truck due to weight limitations. Expected and actual result: -1. **PASS**. No bugs.
- **T002:** The first truck was selected based on distance and capacity. Expected and actual result: 0. **PASS**. No bugs.
- **T003:** The second truck was chosen based on distance. Expected and actual result: 1. **PASS**. No bugs.
- **T004:** The third truck was the best choice based on distance. Expected and actual result: 2. **PASS**. No bugs.
- **T005:** All trucks were equally viable; the first truck was selected. Expected and actual result: 0. **PASS**. No bugs.

The function passed all tests with no bugs identified.

loadShipment

Purpose: This function loads a shipment onto a selected truck and updates the truck's capacity accordingly.

The loadShipment function was evaluated using five test scenarios:

- **T006:** The function was tested with a shipment that should fit into the truck. Expected and actual result: 0 (indicating success). **PASS**. No bugs.
- **T007:** The function was tested with a shipment that exceeds the truck's weight capacity. Expected and actual result: -1 (indicating failure). **PASS**. No bugs.
- **T008:** The function was tested with a shipment that exceeds the truck's volume capacity. Expected and actual result: -1 (indicating failure). **PASS**. No bugs.
- **T009:** A bug was found where the function incorrectly allowed loading even though the truck's weight limit was nearly exceeded. Expected result: -1, but the function

incorrectly returned 0. **FAIL**. **Bug Fixed:** The logic was corrected to properly account for weight limits.

- **T010:** After the bug fix, the function was tested with a shipment that fits with space left. Expected and actual result: 0 (indicating success). **PASS**.

calculateRemainingCapacity

The calculateRemainingCapacity function was tested across five scenarios:

- **T011:** Empty truck. Expected and actual results: remainingVolume = 100.0, remainingWeight = 2500.0. **PASS**.
- **T012:** Partially filled truck. Expected and actual results: remainingVolume = 50.0, remainingWeight = 1500.0. **PASS**.
- **T013:** Truck almost full by weight. Expected and actual results: remainingVolume = 10.0, remainingWeight = 100.0. **PASS**.
- **T014:** A bug was found where the remaining weight was incorrectly calculated. **FAIL**. **Bug Fixed:** Calculation logic corrected.
- **T015:** Full by volume. Expected and actual results: remainingVolume = 0.0, remainingWeight = 1500.0. **PASS**.

isTruckOverload

Purpose: This function checks if the truck is overloaded after loading a shipment.

The isTruckOverload function was evaluated using five test scenarios:

- **T016:** Tested with an empty truck. Expected and actual results: 0 (not overloaded). **PASS**.
- **T017:** Tested with a partially filled truck. Expected and actual results: 0 (not overloaded). **PASS**.
- **T018:** Tested with a truck almost full by weight. Expected and actual results: 0 (not overloaded). **PASS**.
- **T019:** Tested with a truck overloaded by weight. Expected and actual results: 1 (overloaded). **PASS**.
- **T020:** Tested with a truck overloaded by volume. Expected and actual results: 1 (overloaded). **PASS**.

updatePosition

The updatePosition function was tested across five scenarios:

- **T021:** Tested updating the truck's position to a new point. Expected and actual results: {5, 5}. **PASS**.
- **T022:** Tested updating the truck's position to the same point. Expected and actual results: {0, 0}. **PASS**.
- **T023:** Tested updating the truck's position to an adjacent point. Expected and actual results: {1, 1}. **PASS**.
- **T024:** A bug was found where the function failed due to a lack of a NULL pointer check for the truck parameter, causing a runtime error. Expected result: {10, 10}, but the function encountered a runtime error. **FAIL**. **Bug Fixed:** Added a NULL pointer check for the truck parameter.
- **T025:** After the bug fix, the function was tested with a maximum point update. Expected and actual results: {24, 24}. **PASS**.

distance

The distance function was tested across five scenarios:

- **T026:** Tested the distance between two identical points. Expected and actual results: 0.0. **PASS**.
- **T027:** Tested the distance between two adjacent points (0,0) and (1,1). Expected and actual results: 1.41421356237. **PASS**.
- **T028:** Tested the distance between points (0,0) and (3,4). Expected and actual results: 5.0. **PASS**.
- **T029:** Tested the distance between two points in the same row. Expected and actual results: 3.0. **PASS**.
- **T030:** A bug was found where the function incorrectly calculated the distance between two points in the same column. Expected result: 3.0, but the function incorrectly calculated the distance. **FAIL**. **Bug Fixed:** The calculation logic was corrected

WhiteBox Testing Report

findTruckShipment

Test Case IDs: SFT001, SFT002, SFT003, SFT004

Summary of Tests:

SFT001, SFT003, SFT004: All tests passed, confirming correct truck selection.

SFT002: Bug identified and fixed (incorrect truck selection). PASS after fix.

loadShipment

Test Case IDs: SFT001, SFT002, SFT003, SFT004

Summary of Tests:

SFT001, SFT003, SFT004: All tests passed, confirming proper load management.

SFT002: Bug identified and fixed (overloading allowed). PASS after fix.

calculateRemainingCapacity

Test Case IDs: SFT301, SFT302, SFT303, SFT304

Summary of Tests:

SFT301, SFT302, SFT304: All tests passed, confirming accurate capacity calculation.

SFT303: Bug identified and fixed (miscalculated remaining capacity). PASS after fix.

isTruckOverload

Test Case IDs: SFT801, SFT802, SFT803, SFT804

Summary of Tests:

SFT801, SFT802, SFT804: All tests passed, confirming correct overload detection.

SFT803: Bug identified and fixed (failed to detect overload). PASS after fix.

updatePosition

Test Case IDs: SFT001, SFT002, SFT003, SFT004

Summary of Tests:

SFT001, SFT002, SFT004: All tests passed, confirming correct position updates.

SFT003: Bug identified and fixed (NULL pointer crash). PASS after fix.

distance

Test Case IDs: SFT501, SFT502, SFT503, SFT504

Summary of Tests:

SFT501, SFT503, SFT504: All tests passed, confirming accurate distance calculation.

SFT502: Bug identified and fixed (incorrect distance calculation). PASS after fix.

Integration Testing Report

Test Case 1: Valid Input, No Diversion

Input: Weight: 20 units, Volume: 5 units, Target: '12L'

Expected Output: Successful delivery with no diversions required.

Actual Output: The delivery was successfully completed with no diversions. The truck followed the optimal route and reached the destination on time.

Status: PASS

Test Case 2: Valid Input, Minimal Diversion

Input: Weight: 1000 units, Volume: 1 unit, Target: '8Y'

Expected Output: Delivery processed with minimal route changes.

Actual Output: The delivery was processed with a minor diversion due to temporary road conditions. The truck adjusted the route and delivered the shipment without significant delays.

Status: PASS

Test Case 3: Valid Input, Significant Diversion

Input: Weight: 1000 units, Volume: 1 unit, Target: '8Y'

Expected Output: Delivery processed with significant route changes.

Actual Output: Due to unforeseen circumstances, the delivery required significant route changes. The truck adapted to the new route, and the delivery was completed successfully, albeit with some delay.

Status: PASS

Test Case 4: Invalid Destination

Input: Weight: 20 units, Volume: 5 units, Target: '28X'

Expected Output: Error message indicating invalid destination.

Actual Output: The system correctly identified the destination as invalid and provided an appropriate error message, preventing the delivery from proceeding.

Status: PASS

Test Case 5: Invalid Box Size

Input: Weight: 20 units, Volume: 4 units, Target: '12L'

Expected Output: Error message indicating invalid box size.

Actual Output: The system flagged the box size as invalid and displayed an error message, ensuring that the shipment was not loaded onto the truck.

Status: PASS

Acceptance Testing Report

Valid Input for business requirements

Description: Testing the input to determine if it meets business requirements. The following input is expected to exceed the size limit.

Input: Weight: 999 units, Size: 2.3, Destination: '9P'

Expected Output: Invalid size.

Test Results: PASS

Explanation: Program error message matches prediction.

Bug: NO

Input validation must adhere to business requirements (Normal conditions)

Description: Testing the input to ensure it meets business requirements. The following input is expected to trigger an invalid destination error.

Input: Weight: 1000 units, Size: 1.0, Destination: '9Z'

Expected Output: Invalid destination.

Test Results: PASS

Explanation: The program error message matches the predicted outcome.

Bug: NO

Path finding - under capacity business requirement (Normal use case)

Description: Testing if the program can store and pick the shortest path for delivery based on the stored cargo information and truck capacity.

Input: Weight: 200 units, Size: 1.0, Destination: '8Y'

Expected Output: Ship on GREEN LINE, divert: 7U, 7V, 7W, 7X, 7Y, 8Y.

Test Results: PASS

Explanation: Route chosen correctly when truck is at or below capacity.

Bug: NO

Path finding must adhere to business requirements (Maximum use case) and handle over-capacity scenarios effectively.

Description: Testing to determine if the program can recognize when a truck on the shortest path is over capacity and subsequently switch the delivery route to the second shortest.

Input: Weight: 200 units and 801 units, Size: 1.0, Destination: '8Y'

Expected Output: Unexpected results, code did not pass the test.

Test Results: FAILED

Explanation: Unexpected results, code did not pass the test.

Bug: YES

Path finding must handle diversions as part of the business requirements.

Description: Testing to see if the program can determine whether a diversion is necessary to complete a delivery. The following input does not require a diversion.

Input: Weight: 200 units, Size: 1.0, Destination: '12L'

Expected Output: Ship on BLUE LINE, no diversion.

Test Results: PASS

Explanation: Correctly output diversion.

Bug: NO

Path finding must display necessary diversions as part of the business requirements (Normal use case).

Description: Testing to determine if the program can display a set of diversions that correctly leads to the user-input destination.

Input: Weight: 200 units, Size: 3.0, Destination: '8Y'

Expected Output: Ship on GREEN LINE, divert: 7T, 6U, 7V, 7W, 7X, 8Y.

Test Results: PASS

Explanation: The set of diversions received was mostly correct but contained some mistakes.

Bug: YES (small errors)

Diversion accuracy must avoid buildings.

Description: Testing to see if the program can display a set of diversions that correctly leads to the user-input destination while avoiding other buildings on the map.

Input: Weight: 100 units, Size: 0.5, Destination: '25B'

Expected Output: Ship on YELLOW LINE, divert: 19B, 19A, 20A, 21B, 22B, 25B.

Test Results: PASS

Explanation: The set of diversions received was mostly correct until arriving at the destination building, where it didn't fully reach the specific destination point, only the destination building itself.

Bug: YES

Diversion accuracy must handle equal distances.

Description: Checking the program's behavior when a destination is at an equal distance between two routes.

Input: Weight: 100 units, Size: 0.5, Destination: '14W'

Expected Output: Ship on GREEN LINE, divert: 13T, 12U, 11V, 14W.

Test Results: FAILED

Explanation: The program successfully picked one of the routes to deliver the cargo, but a bug was found where the diversion point starts from within a building.

Bug: YES