# Test Description

**Test Name or ID**: BB\_getTruckDistances2

**Test Type**: Black Box

**Description**: To ensure the function calculates and populates the “arr” array correctly, by verifying every index is filled with the truck’s distances and colour codes.

**Setup:**

1. Create a 3x2 double array to store the distances and truck colours (called *distances*).
2. Create a Fleet struct (called *current*) and set the route of each truck in Fleet by calling the get[truck colour]Route() function.
3. Define a Point struct for the destination of the order (called *destination*)
4. Call the getTruckDistances2() function using the arguments mentioned.
5. Create the expected values:
   1. Calculate the distances from the closest points to the destination for each truck using the distance function. Use the trucks’ point structs and the destination point as arguments.
   2. Declare and define 3 new variables to store the colour codes of each truck.
6. Use assertions to compare the expected values and the elements in the 3x2 array. Ensure the array has been properly sorted: The expected colour and distance matches the values in the appropriate indexes.

**Test Function**: BB\_getTruckDistances2

**Test Scenarios:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Test Data | Expected Result | Actual Result | Pass/Fail |
| Check function sorts distances array when the destination coordinates are the same, and each truck’s closest points are valid. | getTruckDistances2 (distances, &current, destination);  destination = { 11, 11 }  Data to calculate expected distances:  …B.route.points = 21  …G.route.points = 15  …Y.route.points = 34 | All 6 assertions return TRUE |  |  |
| Check function sorts distances array when two truck’s closest points are the same. | getTruckDistances2 (distances, &current, destination);  destination = { 6, 4 }  Data to calculate expected distances:  …B.route.points = 8  …G.route.points = 8  …Y.route.points = 9 | All 6 assertions return TRUE |  |  |
| Check function sorts distances array when one of the destination points are 1. | getTruckDistances2 (distances, &current, destination);  destination = { 7, 1 }  Data to calculate expected distances:  …B.route.points = 5  …G.route.points = 5  …Y.route.points = 10 | All 6 assertions return TRUE |  |  |
| Check function sorts distances array when one of the destination coordinates are odd. | getTruckDistances2 (distances, &current, destination);  destination = { 13, 6}  Data to calculate expected distances:  …B.route.points = 23  …G.route.points = 10  …Y.route.points = 12 | All 6 assertions return TRUE |  |  |
| Check function sorts distances array when one of the destination coordinates are both even. | getTruckDistances2 (distances, &current, destination);  destination = { 16, 8}  Data to calculate expected distances:  …B.route.points = 26  …G.route.points = 12  …Y.route.points = 31 | All 6 assertions return TRUE |  |  |
| Check function sorts distances array when all closest points are 20 or above. | getTruckDistances2 (distances, &current, destination);  destination = { 6, 18 }  Data to calculate expected distances:  …B.route.points = 20  …G.route.points = 33  …Y.route.points = 41 | All 6 assertions return TRUE |  |  |
| Check function sorts distances array when one of the destination coordinates and all trucks’ closest points are near the max. | getTruckDistances2 (distances, &current, destination);  destination = { 21, 19 }  Data to calculate expected distances:  …B.route.points = 36  …G.route.points = 36  …Y.route.points = 42 | All 6 assertions return TRUE |  |  |

**Bugs Found**:

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