|  |  |  |  |
| --- | --- | --- | --- |
| Test | Test Data | Expected result | Description |
| Test1 – Calculating distance | struct Point p1 = { 0, 0 };  struct Point p2 = { 3, 4 };  double dist = distance(&p1, &p2);  //Assert  Assert::AreEqual(5.0, dist); | 5.0 | This calculates the distance of the 2 coordinates using the Pythagorean theorem |
| Test 2 – Number of rows and columns | struct Map testMap = populateMap();  int numRows = getNumRows(&testMap);  int numCols = getNumCols(&testMap);  //Assert  Assert::AreEqual(MAP\_ROWS, numRows);  Assert::AreEqual(MAP\_COLS, numCols); | Pass | This checks if the maps rows and columns are initialized correctly |
| Test 3 – Map with a package | struct Map testMap = populateMap();  struct Point package1 = { 5, 5 };  struct Point package2 = { 15, 15 };  struct Point package3 = { 20, 20 };  struct Point package4 = { 8, 8 };  // Calculate shortest paths for each package  struct Route path1 = shortestPath(&testMap, package1, { 10, 10 });  struct Route path2 = shortestPath(&testMap, package2, { 5, 5 });  struct Route path3 = shortestPath(&testMap, package3, { 0, 0 });  struct Route path4 = shortestPath(&testMap, package4, { 15, 15 });  // Check if the path is not empty  Assert::IsTrue(path1.numPoints > 0);  Assert::IsTrue(path2.numPoints > 0);  Assert::IsTrue(path3.numPoints > 0);  Assert::IsTrue(path4.numPoints > 0);    // Check if the number of points in the path is correct  Assert::AreEqual(6, path1.numPoints);  Assert::AreEqual(2, path2.numPoints);  Assert::AreEqual(4, path3.numPoints);  Assert::AreEqual(5, path4.numPoints); | Path1 = 6  Path 2 = 2  Path 3 = 4  Path 4 = 5 | This test has 4 different packages with specific coordinates of the shortest paths and checks if the path have the correct number of points |
| Test 4 – Makes sure there is a destination | struct Map testMap = populateMap();  struct Point package5 = { 0, 0 };  struct Route path5 = shortestPath(&testMap, package5, { 25, 25 });    //Assert  Assert::IsTrue(path5.numPoints == 0); | Path5.numPoints = 0 | This test locates from the start to the end of the map and if the path is equaled to 0 then that means there is a destination for all the paths |
| Test 5 – Correct calculation from start to finish | struct Map testMap = populateMap();  struct Point package6 = { 5, 5 };  struct Route path6 = shortestPath(&testMap, package6, package6);  //Assert  Assert::IsTrue(path6.numPoints == 0); | Path6 = 0 | This test reads as if the package is already at it’s destination and no movement making so that there is no points or 0 |