|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Data** | **Expected Result** | **Description** |
| Test 1 – Testing populating the map | Map map = populateMap();  // Assert  Assert::AreEqual(MAP\_ROWS, getNumRows(&map));  Assert::AreEqual(MAP\_COLS, getNumCols(&map));  Assert::AreEqual(0, map.squares[0][0]); | The MAP\_ROWS and MAP\_COLS matches up with the map and the 0,0 point has a value of 0 | Checking if all the values matches up with the map and the starting point is 0 |
| Test 2 – Adding points | struct Map testMap = populateMap();  Assert::AreEqual(MAP\_ROWS, testMap.numRows);  Assert::AreEqual(MAP\_COLS, testMap.numCols);  Assert::AreEqual(1, testMap.squares[0][4]);  Assert::AreEqual(1, testMap.squares[5][1]);  Assert::AreEqual(0, testMap.squares[2][19]);  Assert::AreEqual(0, testMap.squares[13][20]); | No exact output | This test checks if the map matches with the initialized rows and columns and the map for expected values |
| Test 3 – Adding points | Map map = populateMap();  //Assert  Assert::AreEqual(1, map.squares[2][2]);  Assert::AreEqual(1, map.squares[5][23]); | No exact output | This test verifies whether the map aligns with the expected number of rows and columns and if the initialized map contains the expected values. |
| Test 4 – Adding points | Map map = populateMap();  //Assert  Assert::AreEqual(0, map.squares[15][15]);  Assert::AreEqual(0, map.squares[21][18]); | No exact output | This test ensures that the map dimensions match the expected number of rows and columns, and validates that the initialized map contains the correct values as expected. |