|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Data** | **Expected Result** | **Description** |
| Test5 – Test when the map is populated | Map map = populateMap();  // Check if the map dimensions are correct  Assert::AreEqual(MAP\_ROWS, map.numRows);  Assert::AreEqual(MAP\_COLS, map.numCols);  // Check specific points on the map  Assert::AreEqual(0, map.squares[0][0]);  Assert::AreEqual(1, map.squares[5][5]);  Assert::AreEqual(1, map.squares[10][10]);  Assert::AreEqual(0, map.squares[15][15]);  Assert::AreEqual(1, map.squares[20][20]); | * 0,0 = 0 * 5,5 = 1 * 10,10 = 1 * 15,15 = 0 * 20,20 = 1 | This test checks which parts of the map is populated |
| Test6 – Test when the map is full | Map map = populateMap();  // Check if the map dimensions are correct  Assert::AreEqual(MAP\_ROWS, map.numRows);  Assert::AreEqual(MAP\_COLS, map.numCols);  // Check specific points on the map  for (int i = 0; i < MAP\_ROWS; ++i) {  for (int j = 0; j < MAP\_COLS; ++j) {  Assert::IsTrue(map.squares[i][j] == 0 || map.squares[i][j] == 1);  }  } | Rows = MAP\_ROWS  Cols = MAP\_COLS  Which part of the map is populated or not | Rows and columns should equal the initialized values and listing which parts of the map is a 0 or 1 showing which is populated or not |
| Test7 – Test when the map is empty | Map map = populateMap();  // Check if the map dimensions are correct  Assert::AreEqual(MAP\_ROWS, map.numRows);  Assert::AreEqual(MAP\_COLS, map.numCols);  // Check if all points on the map are empty  for (int i = 0; i < MAP\_ROWS; ++i) {  for (int j = 0; j < MAP\_COLS; ++j) {  Assert::AreEqual(0, map.squares[i][j]);  }  } | Which parts of the map is empty | Similar to last test checking the rows and columns however this time it checks which parts are only empty or 0 |