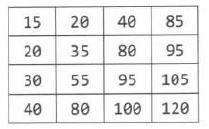
**Question 10.9 – Sorted Matrix:** Given an MX N matrix in which each row and each column is sorted in ascending order, write a method to find an element.

******

|  |
| --- |
| **Part 1 - High Level Algorithm:** Suppose we wanted to search for the 35 in the matrix above.   * We would create two a row iterate and a column iterator. The row iterator would start from the first row and traverse to the last row. The column iterator would traverse from the last row to the first row. * If matrix[row iterator][column iterator] is the item we are looking for, we return true. The item was found. * If matrix[row iterator][column iterator] is greater than the item we are looking for, we decrement the column iterator and loop back. For example, let’s suppose that matrix[row iterator][column iterator] is at the 80. Since 80 is greater than 35, then we decrement the column because it must be somewhere on the left of the row if it exists in this row. * If matrix[row iterator][column iterator] is less than the item we are looking for, we increment the row iterator and loop back. For example, suppose we are at the 20. Either the item is somewhere in the rows below or it is somewhere on the right. It is definitely not in the same row to the left. Since we already check the items on the right (because the column iterator traverse from the right) it must be in the rows below.   **Example Run Through:**   * Initialize the row iterator at the first row (row zero) and initialize the column iterator at the last column (column 3). Since 85 is greater than 35, we decrement the column iterator to two. Since 40 is greater than 35, we decrement the column to 1. * Now we are at row zero, column 1. Since the item at that location (20) is less than 35, we know it is either in the rows below or the items to the left. We know we already checked the items to the left so it must be in the rows below. So we increment the row below. * Once we increment the row, this means the entire rows above have been searched. A row only gets incremented when there is no way the item is in that row. * Now we are at the 35 and we found the item. |

|  |
| --- |
| Part 2 – Implementation:  **public** **static** **boolean** findElement(**int**[][] matrix, **int** elem) {  **int** row = 0; //first row  **int** column = matrix[0].length - 1; //last column    **while** (!*outOfBounds*(matrix, row, column)) {  **if** (matrix[row][column] == elem) {  **return** **true**;//found the item  } **else** **if** (matrix[row][column] > elem) {  column--;  } **else** {  row++;  }  }  **return** **false**;  }    **public** **static** **boolean** outOfBounds(**int**[][] matrix, **int** row, **int** column){  **return** (row >= matrix.length || column < 0);  }  Also keep searching until you reach a point where it is “out of bounds”. This occurs in any of two cases:   1. Your row iterator increments passed the last row. Remember, whenever you increment the row, it means that the search item is definitely not in the rows above. When you increment passed the last row, it means the search item is definitely not in the entire matrix. 2. Your column iterator decrements passed the first row. This means that the search item is not in the current row (you just searched it entirely) and it is not in any of the rows below it (because all the items in the rows below are guaranteed to be larger than the item you just searched). For example, suppose you are searching for 17 and you are at the 20. Since 20 is greater than 17, you decrement the column past the first column. Every element below 20 is greater than 20 and every element right of 20 is greater than 20 and every element above 20 has already been searched so you can conclude 17 is not in the array at all. |