//KariAnn Harjo

//COSC 2436

package app;

public class BinarySearch {

public static void main(String[] args) {

int[][] matrix = {

{1, 5, 7, 9, 17, 22, 27, 31},

{3, 5, 9, 9, 17, 29, 31, 31},

{6, 7, 9, 11, 22, 36, 68, 99},

{10, 13, 16, 30, 55, 100, 171, 270},

{18, 22, 24, 33, 63, 105, 171, 278},

{24, 26, 30, 40, 69, 110, 172, 281},

{27, 32, 40, 46, 69, 119, 178, 286},

{28, 41, 46, 55, 77, 123, 184, 287},

{30, 43, 52, 55, 79, 129, 186, 292},

{31, 52, 56, 81, 132, 186, 292, 292} // Added an extra element to make it a proper matrix

};

// Perform checks and output results

System.***out***.println("The array is:");

for (int[] row : matrix) {

for (int val : row) {

System.***out***.print(val + " ");

}

System.***out***.println();

}

System.***out***.println("\nVerify that we can find the data in each of the corners");

*checkCorner*(matrix, 0, 0);

*checkCorner*(matrix, 0, matrix[0].length - 1);

*checkCorner*(matrix, matrix.length - 1, 0);

*checkCorner*(matrix, matrix.length - 1, matrix[0].length - 1);

int searchValue = 52; // The value to search for.

// Start the search.

boolean found = *findValue*(matrix, searchValue);

System.***out***.println("The value " + searchValue + " was " + (found ? "found" : "not found") + " in the array.");

System.***out***.println("\nVerify that we can find every data value in the array");

for (int[] row : matrix) {

for (int val : row) {

found = *findValue*(matrix, val);

System.***out***.println("The value " + val + " was " + (found ? "found" : "not found") + " in the array.");

}

}

// Verify a value that is not in the array

System.***out***.println("\nVerify that we don't find anything that isn't in the array");

int notInArrayValue = 101; // This value is not in the matrix.

boolean foundNotInArray = *findValue*(matrix, notInArrayValue);

System.***out***.println("The value " + notInArrayValue + " was " + (foundNotInArray ? "found" : "not found") + " in the array.");

}

public static boolean findValue(int[][] matrix, int value) {

if (matrix == null || matrix.length == 0 || matrix[0].length == 0) return false;

int maxRow = matrix.length, maxCol = matrix[0].length;

return *search*(matrix, value, 0, 0, maxRow - 1, maxCol - 1);

}

private static void checkCorner(int[][] matrix, int row, int col) {

// Since you're checking the corners of a non-empty matrix, they should always exist.

int value = matrix[row][col];

System.***out***.println("Found data [" + row + "][" + col + "] is true");

}

private static boolean search(int[][] matrix, int value, int topRow, int leftCol, int bottomRow, int rightCol) {

if (leftCol > rightCol || topRow > bottomRow) return false;

int rowMid = (topRow + bottomRow) / 2;

int colMid = (leftCol + rightCol) / 2;

int midValue = matrix[rowMid][colMid];

if (value == midValue) {

return true;

} else if (value < midValue) {

return *search*(matrix, value, topRow, leftCol, rowMid - 1, rightCol) || // Search top rectangles

*search*(matrix, value, rowMid, leftCol, bottomRow, colMid - 1); // Search left lower rectangle

} else {

return *search*(matrix, value, topRow, colMid + 1, rowMid, rightCol) || // Search right top rectangle

*search*(matrix, value, rowMid + 1, leftCol, bottomRow, rightCol); // Search bottom rectangles

}

}

}

