Fun with 2D arrays

Path Walk:

Write a program that conducts a "Path Walk" on a 2-dimensional grid. The size of the grid is n by n, n <= 10 with (0, 0) in the Upper Left hand corner. The walk should begin at square `row = startRow` and `col = startCol` and follows the Path determined by walk, an array of `ints`. Each step consists of moving one square vertically, horizontally, or diagonally (thus, there are eight possible choices for a next move) as determine defined in the following table. The walker should be able to "wrap around" the square. For instance, if the walker is currently in a leftmost square, and the next step choice is directly to the left, the walker ends up in the corresponding rightmost square. If the walker is in a topmost square, and the next step choice is up and to the right, the walker ends up one square to the right of the corresponding bottommost square. You should record how many times each square is visited and output the result.

`0 = up`	`1 = up and right`	`2 = right`	`3 = right and down`
`4 = down`	`5 = down and left`	`6 = left;`	`7 = up and left`

The `PathWalk(size, startRow, startCol, walk)` method returns a `int[ ][ ]` containing how many times each square is visited.

- Remember square (0, 0) in the Upper Left hand corner. Therefore move 3 (right and down) increases both row and column.

The following table shows results of a `PathWalk` with `size = 4`, `startRow = 2`, `startCol = 1` and with `walk = { 3, 6, 5, 2, 1, 0, 7, 4, 3}`

		columns			
		0	1	2	3
rows	0	1	1	0	0
	1	0	1	0	0
	2	0	2	1	0
	3	0	1	3	0

The following code shows the results of the `PathWalk` method.

The following code	Returns

`FunWith2DArrays f2d = new FunWith2DArrays();`
`int[] walk = { 3, 6, 5, 2, 1, 0, 7, 4, 3};`
`int[][] ans = FunWith2DArrays.pathWalk(4, 2, 1, walk);`	

`ans[0][0]`	1
`ans[0][1]`	1
`ans[0][2]`	0
`ans[0][3]`	0
`ans[1][0]`	0
`ans[1][1]`	1
`ans[1][2]`	0

Fun with 2D arrays

`ans[1][3]`	0
`ans[2][0]`	0
`ans[2][1]`	2
`ans[2][2]`	1
`ans[2][3]`	0
`ans[3][0]`	0
`ans[3][1]`	1
`ans[3][2]`	3
`ans[3][3]`	0

Gynn Numbers:

A number is said to have Gynn property if the number looks like an upside-down `String` of letters. For this problem, the following numbers look like the corresponding upside-down letters.

0	O	5	S
1	I	6	g
3	E	7	L
4	h	8	B

In short, if a number contains only the digits 0, 1, 3, 4, 5, 6, 7, and 8, the number is said to have the Gynn property.

The following code shows the results of the `hasGynnProperty` method.

The following code	Returns
`FunWith2DArrays f2d = new FunWith2DArrays();` `FunWith2DArrays.hasGynnProperty(41587));`	true
`FunWith2DArrays.hasGynnProperty(2587)`	false

Gynn Arrays:

A 2-dimensional array is said to have the Gynn property if either:

- All rows contain more numbers with the Gynn property than numbers that do not.
- **Or** all columns contain more numbers with the Gynn property than numbers that do not.

The following code shows the results of the `hasArrayGynnProperty` method.

The following code	Returns
`FunWith2DArrays f2d = new FunWith2DArrays();`	

`int[][] gynnNumbers1 = { { 3, 6, 5, 2},`
`{ 1, 0, 7, 49},`
`{23, 66, 10, 88},`
`{48, 53, 200, 308} };`
	true

Fun with 2D arrays

`FunWith2DArrays.hasArrayGynnProperty(gynnNumbers1)`	

`int[][] gynnNumbers2 = { { 39, 6, 5, 8},`
`{ 1, 0, 72, 49},`
`{ 93, 66, 1, 88},`
`{428, 53, 0, 308} };`
`FunWith2DArrays.hasArrayGynnProperty(gynnNumbers2)`
	`false`

A 2-dimensional array is said to have the Super Gynn property if both all rows contain more numbers with the Gynn property than numbers that do not have the Gynn property **and** all columns contain more numbers with the Gynn property than numbers that do not have the Gynn property.

The following code shows the results of the `hasArraySuperGynnProperty` method.

The following code	Returns
`FunWith2DArrays f2d = new FunWith2DArrays();`	
`int[][] superGynnNumbers1 = { {3, 659, 5, 17}, {1, 0, 7, 49}, {23, 66, 10, 88}, {48, 53, 200, 308} };` `FunWith2DArrays.hasArraySuperGynnProperty(superGynnNumbers1)`	`true`

`int[][] superGynnNumbers2 = { { 3, 659, 5, 127},`
`{ 1, 0, 7, 49},`
`{23, 66, 10, 88},`
`{48, 53, 200, 308} };`
`FunWith2DArrays.hasArraySuperGynnProperty(superGynnNumbers2)`
	`false`