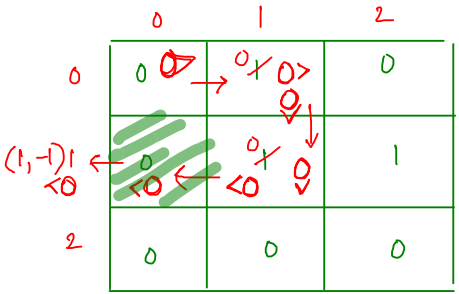
Q: Find the way:



★ always the mouse tries to move forward,
If it sees a
0 - forward
1 - turn right, forward

0> (right) = 0 <0 (left) = 2

0 (down) = 1 0 (top) = 3

★ facing = (facing + m[i][j]) % 4

```
if (facing == 0) {
    j++;
}
else if (facing == 1) {
    i++;
}
else if (facing == 2) {
    j--;
}
else {
    i--;
}
```

(i,j) 0> → (i,j+1) 0> <0 ← <0 (i,j)
(i,j-1)

(i,j) 0 ↓ (i+1,j) 0

0 (i-1,j) ↑ 0 (i,j)

0	>
1	V
2	<
3	∧

f = $\overline{0\ 1\ 2\ 3}\ \overline{0\ 1\ 2\ 3}\ \overline{0\ 1\ 2\ 3}$
(cyclic pattern) (%)

Q: Maxima and minima :

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

minR0 = 1 minR0 = 1

maxC0 = 7 maxC1 = 8

minR2 = 7

maxC0 = 7

* goto every element and check whether it is the minimum ele of the Row and maximum ele of Col

minR0 = 1

maxC2 = 9

```
for( let i = 0; i < rows; i++) {
    for( let j = 0; j < cols; j++) {
        let minR = minRow(mat, i);
        let maxC = maxCol(mat, j);
        if (mat[i][j] == minR &&
            mat[i][j] == maxC ) {
            return mat[i][j];
        }
    }
}
return -1;
```

cols

rows + cols

rows

SC: O(1)

TC: O(rows * cols * (rows + cols))

if rows == cols == n

$\Rightarrow O(n * n * 2n) \Rightarrow O(2n^3)$

$\Rightarrow O(n^3)$

Optimize / Improve : (precomputation technique)

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

→ minEleRow

0	1	2
1	4	7

(rows)

0	1	2
7	8	9

(cols)

→ maxEleCol

O(rows * cols)
for precomputation

```
for(let i=0; i<rows; i++) {
  for(let j=0; j<cols; j++) {
    let minR = minEleRow[i];
    let maxC = maxEleCol[j];   } O(1)
    if (mat[i][j] == minR &&
        mat[i][j] == maxC) {
        return mat[i][j];
    }
  }
}
```

TC : precomputation + logic

$O(row * cols + rows * cols) \Rightarrow O(n^2 + n^2) \Rightarrow O(2n^2)$

if rows = cols = n $\Rightarrow O(n^2) \downarrow$ SC : $O(rows + cols) \Rightarrow O(2n) \Rightarrow O(n) \uparrow$

```
function maximaMinima(mat) {
  const rows = mat.length;
  const cols = mat[0].length;

  const minEleRow = [];
  for (let i = 0; i < rows; i++) {
    minEleRow[i] = Infinity;
  }
  const maxEleCol = [];
  for (let i = 0; i < cols; i++) {
    maxEleCol[i] = -Infinity;
  }

  for (let i = 0; i < rows; i++) {
    for (let j = 0; j < cols; j++) {
      if (mat[i][j] < minEleRow[i]) {
        minEleRow[i] = mat[i][j];
      }
      if (mat[i][j] > maxEleCol[j]) {
        maxEleCol[j] = mat[i][j];
      }
    }
  }

  for (let i = 0; i < rows; i++) {
    for (let j = 0; j < cols; j++) {
      const minR = minEleRow[i];
      const maxC = maxEleCol[j];
      if (mat[i][j] == minR && mat[i][j] == maxC) {
        return mat[i][j];
      }
    }
  }
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

0	1	2	
~~∞~~	~~∞~~	~~∞~~	min ele Row
1	4	7	

0	1	2	
~~-∞~~	~~-∞~~	~~-∞~~	max ele Col
~~1~~ ~~4~~ 7	~~2~~ ~~5~~ 8	~~3~~ ~~6~~ 9	

minEleRow[0] → min val in 0th Row

maxEle Col [1] → max val in 1st col

① 1 < ∞
1 > -∞

② 2 < 1
2 > -∞

③ 3 < 1
3 > -∞

④ 4 < ∞
4 > 1

⑤ 5 < 4
5 > 2

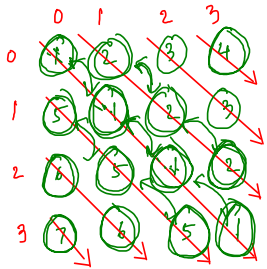
⑥ 6 < 4
6 > 3

⑦ 7 < ∞
7 > 4

⑧ 8 < 7
8 > 5

⑨ 9 < 7
9 > 6

Q: Toeplitz Matrix :



only one ele
no ele's to compare

$j < 4-1$
$j < 3$

$(i-1, j-1) \leftarrow (i, j) \rightarrow (i+1, j+1)$

corresponding diagonal element

```
for( let i = 0; i < rows-1; i++) {
    for( let j = 0; j < cols -1; j++) {
        if ( mat[i][j] != mat[i+1][j+1]) {
            return false;
        }
    }
}
return true;
```

7 : (3, 0)

6 : (2, 0) , (3, 1)

5 : (1, 0) , (2, 1) , (3, 2)

1 : (0, 0) , (1, 1) , (2, 2) , (3, 3)

2 : (0, 1) , (1, 2) , (2, 3)

3 : (0, 2) , (1, 3)

4 : (0, 3)

① i = 0
- j = 0 → (0, 0) (1, 1)
- j = 1 → (0, 1) (1, 2)
- j = 2 → (0, 2) (1, 3)
- j = 3 ✗

② i = 1;
- j = 0 → (1, 0) (2, 1)
- 1 → (1, 1) (2, 2)
- 2 → (1, 2) (2, 3)
- 3 ✗

③ i = 2
- j = 0 → (2, 0) (3, 1)
- 1 → (2, 1) (3, 2)
- 2 → (2, 2) (3, 3)
- 3 ✗

④ i = 3 ✗ $j < 4-1$, $i < 3$

1. what did you like the most
2. what did you hate the most
3. one/two word about the module

Imp Methods / Techniques {

running stream
precomputation
two pointer
rotation (cyclic → %)
generating pairs, triplets, subarrays

}