Labyrinth

Given an NxN, N>0 matrix that represents a labyrinth (i.e., maze), find a path from the upper left corner (i.e., index <0, 0>) to the lower right corner (<N-1, N-1>).

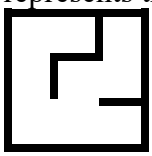
There is a wall around the perimeter of the maze.

Each element of the matrix (i.e., location in the labyrinth) describes the presence or absence of the right and lower walls as follows:
0 – no right wall, no lower wall
1 – no right wall, has lower wall
2 – has right wall, no lower wall
3 – has right wall, has lower wall

For example, this 3x3 matrix:

0	3	2
2	0	3
1	1	3

represents this labyrinth:



Write a function `labyrinth()` that finds a path through a labyrinth represented by an NxN matrix `l[][]`.
Indicate the path you found using an NxN matrix `bool p[][]` where `p[i][j]=true` if and only if it is part of the path you found.

```
void labyrinth(vector<vector<int>> &l, vector<vector<bool>> &p)
```

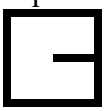
where
`l` – NxN matrix describing the labyrinth
`p` – NxN `bool` matrix describing your path. It will be passed to you initialized as follows:
`p[0][0]=true` (to indicate that all paths start there),
all other elements `p[i][j]=false`

Examples:

Input:
`l[][]`

0	3
1	3

Representing:



Output:
`p[][]`

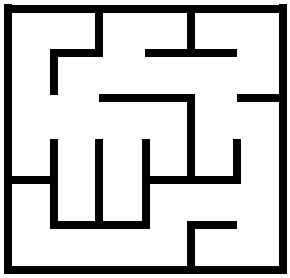
1	0
1	1

Representing:

Input:
`l[][]`

0	3	0	3	1	2
2	0	1	1	0	3
0	0	0	2	0	2
3	2	2	3	3	2
2	3	3	0	1	2
1	1	1	3	1	3

Representing:



Output:
`p[][]`

1	0	0	0	0	0
1	1	1	1	1	0
1	1	0	0	1	1
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1

Representing: