Maze

The maze is basically a field that consists of squares that can and cannot be walked on.
It's represented by an array of lines, each line is an array of cells.
A cell can be addressed as `maze[lineIndex][columnIndex]`. Coordinates start from the top-left cell `{ x: 0, y: 0 }`.
A walk-ability of a cell is determined by a boolean value: `false` means a wall, `true` means a passage.

```
const topLeft = { x: 0, y: 0 };
console.log(maze[topLeft.y][topLeft.x]); // false (top-left cell isn't
walkable)
```

One point at the edge is the entry point. In the example it's `{ x: 0, y: 3 }` (line #4, column #1).
Any other point at the edge that can be reached by walking from the entry point, can be called an exit point.
The example maze has one exit at `{ x: 4, y: 0 }`.

Creating a path from the entry point to the exit point is the test task.
As a result, we expect a function that accepts a maze and a starting coordinates and returns a path that leads from starting point to the exit. The format of the return value is up to you.
Below the example maze, you can find some typescript that can help to understand the expected result function signature.

Example maze:

```
const X = false; // wall
const _ = true; // pass
const maze = [
    [X,X,X,X,_,X,X,X,X],
    [X,_,X,_,_,X,_,_,X],
    [X,_,X,X,_,X,_,X,X],
    [_,_,X,_,_,_,_,X,_],
    [X,_,X,_,X,_,X,X,X],
    [X,_,_,_,X,_,_,_,X],
    [X,X,X,X,X,X,X,X,X],
];
```

Expected result:

```
interface Coords {
  x: number;
  y: number;
}
type Maze = boolean[][];
function walk(maze: Maze, start: Coords): unknown;
```