Please ensure you read the question carefully and solve it using javascript.

```
// 1. Find the number of islands
// Given a boolean 2D matrix, find the number of islands. A group of connected 1s
forms an island. For example, the below matrix contains 5 islands

// const matrix = [
//   [1, 1, 0, 0, 0],
//   [0, 1, 0, 0, 1],
//   [1, 0, 0, 1, 1],
//   [0, 0, 0, 0, 0],
//   [1, 0, 1, 0, 1]
// ];

// Output: 5
```

```
// 2. In this kata, you must create a digital root function using recursion or with
time complexity of O(1).

// A digital root is the recursive sum of all the digits in a number. Given n, take
the sum of the digits of n. If that value has more than one digit, continue reducing
in this way until a single-digit number is produced. This is only applicable to the
natural numbers.

// Here's how it works:

// digital_root(16)
// => 1 + 6
// => 7

// digital_root(942)
// => 9 + 4 + 2
// => 15 ...
// => 1 + 5
// => 6
```

```
// 3. John and Mary want to travel between a few towns A, B, C ... Mary has on a sheet
of paper a list of distances between these towns. ls = [50, 55, 57, 58, 60]. John is
```

Please ensure you read the question carefully and solve it using javascript.

```
tired of driving and he says to Mary that he doesn't want to drive more than t = 174
miles and he will visit only 3 towns.

// Which distances, hence which towns, they will choose so that the sum of the
distances is the biggest possible - to please Mary - but less than t - to please John-
?

// Example:

// With list ls and 3 towns to visit they can make a choice between:
[50,55,57],[50,55,58],[50,55,60],[50,57,58],[50,57,60],[50,58,60],[55,57,58],[55,57,60
],[55,58,60],[57,58,60].

// The sums of distances are then: 162, 163, 165, 165, 167, 168, 170, 172, 173, 175.

// The biggest possible sum taking a limit of 174 into account is then 173 and the
distances of the 3 corresponding towns is [55, 58, 60].

// The function chooseBestSum will take as parameters t (maximum sum of distances,
integer >= 0), k (number of towns to visit, k >= 1) and ls (list of distances, all
distances are positive or null integers and this list has at least one element). The
function returns the "best" sum ie the biggest possible sum of k distances less than
or equal to the given limit t, if that sum exists, or otherwise return null.

// Examples:

// ts = [50, 55, 56, 57, 58], chooseBestSum(163, 3, ts) -> 163
// xs = [50], chooseBestSum(163, 3, xs) -> null
// ys = [91, 74, 73, 85, 73, 81, 87], chooseBestSum(230, 3, ys) -> 228
```

```
// 4. Short Intro - Some of you might remember spending afternoons playing Street
Fighter 2 in some Arcade back in the 90s or emulating it nowadays with the numerous
emulators for retro consoles.
// You'll have to simulate the video game's character selection screen behaviour, more
specifically the selection grid. Such screen looks like this:
```

Please ensure you read the question carefully and solve it using javascript.

```
// Selection Grid Layout in text:
// | Ryu   | E.Honda | Blanka  | Guile   | Balrog | Vega    |
// | Ken   | Chun Li | Zangief | Dhalsim | Sagat  | M.Bison |

// Input
// the list of game characters in a 2x6 grid;
// the initial position of the selection cursor (top-left is (0,0));
// a list of moves of the selection cursor (which are up, down, left, right);

// Output
// the list of characters who have been hovered by the selection cursor after all the
moves (ordered and with repetition, all the ones after a move, whether successful or
not, see tests);

// Rules
// Selection cursor is circular horizontally but not vertically!

// As you might remember from the game, the selection cursor rotates horizontally but
not vertically; that means that if I'm in the leftmost and I try to go left again I'll
get to the rightmost (examples: from Ryu to Vega, from Ken to M.Bison) and vice versa
from rightmost to leftmost.
// Instead, if I try to go further up from the upmost or further down from the
downmost, I'll just stay where I am located (examples: you can't go lower than lowest
row: Ken, Chun Li, Zangief, Dhalsim, Sagat and M.Bison in the above image; you can't
go upper than highest row: Ryu, E.Honda, Blanka, Guile, Balrog and Vega in the above
image).

// Examples

// 1.
// fighters = [
//     ["Ryu", "E.Honda", "Blanka", "Guile", "Balrog", "Vega"],
//     ["Ken", "Chun Li", "Zangief", "Dhalsim", "Sagat", "M.Bison"]
// ]
// initial_position = (0,0)
// moves = ['up', 'left', 'right', 'left', 'left']
// Result: ['Ryu', 'Vega', 'Ryu', 'Vega', 'Balrog']
// as the characters I've been hovering with the selection cursor during my moves.
Notice: Ryu is the first just because it "fails" the first up See test cases for more
examples.
```

Please ensure you read the question carefully and solve it using javascript.

```
// 2.
// fighters = [
//     ["Ryu", "E.Honda", "Blanka", "Guile", "Balrog", "Vega"],
//     ["Ken", "Chun Li", "Zangief", "Dhalsim", "Sagat", "M.Bison"]
// ]
// initial_position = (0,0)
// moves = ['right', 'down', 'left', 'left', 'left', 'left', 'right']
// Result: ['E.Honda', 'Chun Li', 'Ken', 'M.Bison', 'Sagat', 'Dhalsim', 'Sagat']
```