# Problem 5 – Game of Life

The game of life is a simple game in which the player sets the initial state of the board **(always with size 10x10)** and the computer calculates the next states of the board, by following four simple rules.

1. Any live cell with fewer than two live neighbours dies.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies due to overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

A live cell is a cell which has value of '1', dead cell is a cell which has value of '0'. By default all cells, except the ones specified in the input, are dead.

**Note that all calculations need to happen simultaneously! See the example.**

Your task is to **generate the next state of the board and the print it to the console**.

## Input

The input data should be read from the console. It consists of several input values, each at a separate line:

- Number of input coordinates (x, y) **n**: how many coordinates will be entered.
- Coordinates: **x** and **y** (**each at a separate line**), which **set the cell [x, y] to 1**.
- Rows are counted from top to bottom (0 to 9) and columns are counter from right to left (0 to 9).

The input data will always be valid and in the format described. There is no need to check it explicitly.

## Output

- The output data must be printed on the console.
- You must calculate the next generation and then print the resulting board.

## Constraints

- **n** is an integer number in range [0 ... 2 147 483 647].
- **x** and **y** are integer numbers in range [0…9].
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

## Examples

| Input | Output | Comments |
|---|---|---|
| 3 | 0000000000 | `n = 3 => we'll receive 3 pairs` **`(x, y)`**`. We set the bits to 1 at these` |
| 1 | 0000000000 | `positions so the board looks like this:` |
| 6 | 0011100000 | `0000000000` |
| 2 | 0000000000 | `0001000000` |
| 6 | 0000000000 | `0001000000` |
| 3 | 0000000000 | `0001000000` |
| 6 | 0000000000 | `0000000000` |
|  | 0000000000 | `0000000000` |
|  | 0000000000 | `0000000000` |
|  | 0000000000 | `0000000000` |
|  |  | `0000000000` |
|  |  | `0000000000` |
|  |  |  |
|  |  | The **shaded zeros** have **exactly 3 live neighbours**, so in the next generation they will become ones. The **bolded one between them** has **2 live neighbours**, so it will **stay alive**. The **other ones** each have only **one live neighbour**, so they will **become zeros**. |