Problem 2 - Bitwise Minesweeper

Remember Stamat? Well, he was the one who abandoned you on the last teamwork when you were creating the Bitwise 1024 game. Well, on your next teamwork you might be more fortunate right? So this time, you are in a team with Marin and he was really eager to create a bitwise version of the minesweeper game so you agreed to do it. You can't afford him to get sad and abandon your team as Stamat did right? Well unfortunately, Marin had really important things to do like playing League of Legends all day long so guess what – You got abandoned on this teamwork as well. So, get a tissue, wipe those tears off and get the work done. You have 6 hours until the teamwork defense.

On the first input line, you are given a number **N** (0 < **N** < 21) and on the next **N** lines you are given numbers representing the minefield of the game. The **ones** represent mines and the **zeroes** represent available places to move. You are given a player who starts at **the upper right corner** of the board and is also represented by a **one** (The first number will always contain a one at the 0 position). On the next lines, you are given commands which are either **left, right, up or down**. When you receive any of these commands, you have to move the player in the desired direction by one space. If on the space you are moving to is a one, **the player has stepped on a bomb** and the game is over. Furthermore, if the player is located at **any of the edges of the board** and moves towards that edge, his next position should be on **the other side of the board**.

When you receive the command **end** OR when your player hits a mine you should end the game. In the end of the game, you should print out the current state of the numbers representing the board.

Input

The input will be read from the standard input.

- On the **first line** you will be given an integer **N** – **the number of input numbers.**
- On the next **N** lines, you will be given numbers to be processed as a game field.
- On the next lines, you will receive commands which determine how the state of the game will change. They will be either **left, right, up or down**.
- When the player steps on a mine or when you receive the command **end**, proceed to the output of the program.

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

Output

The output should be printed on the standard output.

- If the game has ended due to the player stepping on a mine, you should print the message **GAME OVER. Stepped a mine at X Y** where **X** is the current row of the player and **Y** is the current column of the player (Note: the columns are processed from right to left.)
- After the game has ended, you should print out the current state of the numbers representing the game board.
- **Note**: The numbers outputted should be regarded as **unsigned integers**.

Constraints

- The number N will be in the range [1 … 20].
- The input numbers will be in the range [0 … 2^{32}]
- The number of **commands** will be in the range [1 … 30].
- Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input	Output	Comments
2 31 50 right down left left right up end	GAME OVER. Stepped a mine at 1 1 30 51	00000000000000000000000000011111 right 00000000000000000000000000110010 ▼ 10000000000000000000000000011110 down 00000000000000000000000000110010 ▼ 00000000000000000000000000011110 left 10000000000000000000000000110010 ▼ 00000000000000000000000000011110 left 00000000000000000000000000110011 ▼ 00000000000000000000000000011110 game over 00000000000000000000000000110011

Input	Output	Comments
3 69 26 34 right up right right up left left end	68 2147483674 34	00000000000000000000000001000101 00000000000000000000000000011010 00000000000000000000000000100010 ▼ 10000000000000000000000001000100 00000000000000000000000000011010 00000000000000000000000000100010 ▼ 00000000000000000000000001000100 00000000000000000000000000011010 10000000000000000000000000100010 ▼ 00000000000000000000000001000100 00000000000000000000000000011010 01000000000000000000000000100010 ▼ 00000000000000000000000001000100 00000000000000000000000000011010 00100000000000000000000000100010 ▼ 00000000000000000000000001000100 00100000000000000000000000011010 00000000000000000000000000100010 ▼ 00000000000000000000000001000100 01000000000000000000000000011010 00000000000000000000000000100010 ▼ 00000000000000000000000001000100 10000000000000000000000000011010 00000000000000000000000000100010