

## Problem 3 – Labyrinth Dash

Enough hard problems. Let's play a game! You will be given the layout of a labyrinth (a two-dimensional array) and a series of moves. Your task is to navigate the labyrinth and **print the outcome of each move**.

On the first line of input you will be given the **number N**, representing the **count of rows** of the labyrinth. On each of the next N lines you will receive a **string, containing the layout of the given row**. On the last line of input you will receive a **string, containing the moves** you need to make. Each move will be one of the following symbols: **"v"** (move down), **"^"** (move up), **"<"** (move left) or **">"** (move right). The **player starts with 3 lives and begins the journey at position (0, 0)**. When you make a move, there can be several different outcomes:

- 1) **Hit a wall** – a wall is represented by the symbols **"\_"** (underscore) and **"|"** (pipe). Hitting a wall means the player stays in place; in this case you should print on the console **"Bumped a wall."**
- 2) **Land on an obstacle** – obstacles are the following symbols: **"@"**, **"#"**, **"\*"**. If you move to a position containing one of these symbols the player loses a life point and you should print **"Ouch! That hurt! Lives left: X"** on the console. If the player is left with 0 lives, the game ends and you should print **"No lives left! Game Over!"**
- 3) **Get a new life** – when you land on the symbol **"\$"** the player receives an additional life point. Print **"Awesome! Lives left: X"** on the console. Additional lives ('\$') are removed once the player passes through the cell (i.e. they are replaced with dots).
- 4) **Drop out of the labyrinth** – if you land on an empty cell (one containing a space), or outside the boundaries of the array, the game ends and you should print **"Fell off a cliff! Game Over!"**
- 5) **Land on the symbol "."** (dot) – move the player to the new position, nothing else happens; print on the console **"Made a move!"**

When the game ends (either the player has lost or all moves were made), print **"Total moves made: X"**.

### Input

- The input data should be read from the console.
- On the first line of input you will receive the number N – number of rows of the labyrinth.
- On the next N lines you will receive the layout of the labyrinth.
- On the last line you will receive the moves you need to make as a string.
- The input data 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 console.
- For each outcome print the required output as described above.

### Constraints

- The number N will be an integer in the range [1 ... 15].
- The labyrinth will contain only the symbols – **"\_"**, **"|"**, **"@"**, **"#"**, **"\*"**, **"\$"**, **" "** (single whitespace), **"."**.
- The string containing the moves will contain only the symbols – **"v"**, **"^"**, **"<"**, and **">"**.
- Allowed working time for your program: 0.5 seconds. Allowed memory: 16 MB.

### Examples

Input	Output	Comments
5 .  ..  *.\$ . ###... _____ >v>>vv>>>^^^<<	Bumped a wall. Made a move! Made a move! Bumped a wall. Made a move! Ouch! That hurt! Lives left: 2 Ouch! That hurt! Lives left: 1 Made a move! Made a move! Fell off a cliff! Game Over! Total moves made: 8	Player starts at (0, 0). First move is ">" (right), which takes the player into a wall. Next, he moves down and right. The next move is right again and he hits another wall. He then moves down twice, on the second move he lands on an obstacle ("#") and loses a life point. He then moves right and loses another life. Two moves to the right are followed by a move upwards which takes him out of the labyrinth (empty cell), so the game is over. The total number of moves where the player actually changed position is 8.

