# Problem 3248: Snake in Matrix

## Problem Information

**Difficulty:** Easy
**Acceptance Rate:** 82.13%
**Paid Only:** No
**Tags:** Array, String, Simulation

## Problem Description

There is a snake in an `n x n` matrix `grid` and can move in **four possible directions**. Each cell in the `grid` is identified by the position: `grid[i][j] = (i * n) + j`.

The snake starts at cell 0 and follows a sequence of commands.

You are given an integer `n` representing the size of the `grid` and an array of strings `commands` where each `command[i]` is either `"UP"`, `"RIGHT"`, `"DOWN"`, and `"LEFT"`. It's guaranteed that the snake will remain within the `grid` boundaries throughout its movement.

Return the position of the final cell where the snake ends up after executing `commands`.

**Example 1:**

**Input:** n = 2, commands = ["RIGHT","DOWN"]

**Output:** 3

**Explanation:**

0 | 1 ---|--- 2 | 3 0 | 1 ---|--- 2 | 3 0 | 1 ---|--- 2 | 3 **Example 2:**

**Input:** n = 3, commands = ["DOWN","RIGHT","UP"]

**Output:** 1

**Explanation:**

0 | 1 | 2 ---|---|--- 3 | 4 | 5 6 | 7 | 8 0 | 1 | 2 ---|---|--- 3 | 4 | 5 6 | 7 | 8 0 | 1 | 2 ---|---|--- 3 | 4 | 5 6 | 7 | 8 0 | 1 | 2 ---|---|--- 3 | 4 | 5 6 | 7 | 8

**Constraints:**

* `2 <= n <= 10` * `1 <= commands.length <= 100` * `commands` consists only of `"UP"`, `"RIGHT"`, `"DOWN"`, and `"LEFT"`. * The input is generated such the snake will not move outside of the boundaries.

## Code Snippets

**C++:**

```
class Solution {
public:
int finalPositionOfSnake(int n, vector<string>& commands) {

}
};
```

**Java:**

```
class Solution {
public int finalPositionOfSnake(int n, List<String> commands) {

}
}
```

**Python3:**

```
class Solution:
def finalPositionOfSnake(self, n: int, commands: List[str]) -> int:
```