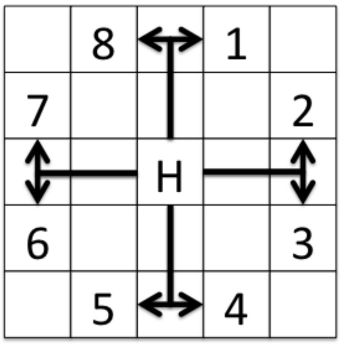
Problem 2 – Horsy

You are given a matrix of directions. **The rows are zero-based.** The possible directions are eight and follow the horse moves from the game chess. The directions are as follows:



You should generate another matrix with the same size. The numbers in each row are consecutive and decreasing. The leftmost (first) number in each row is a power of 2, calculated with the formula **2^{row}** where **row** is the number of this row and each cell to the right on the same row is lesser with 1.

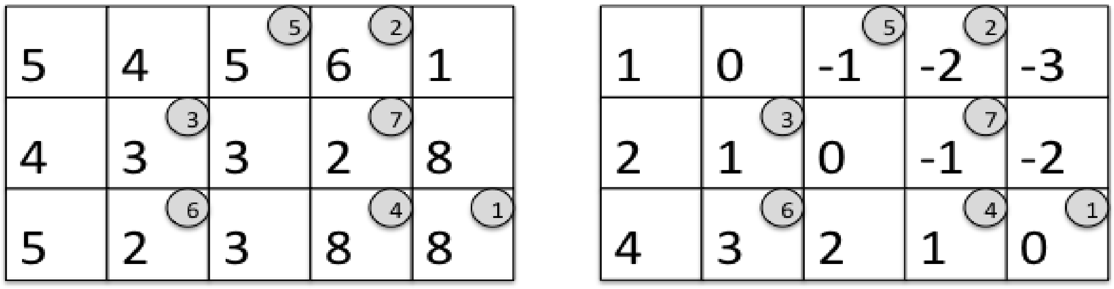
Your task is to find the sum that can be found, using the given directions and summing the numbers in each cell you step into. If with the current direction, you step out of the matrix – print "**Go go Horsy! Collected SUM weeds'**", where SUM is the collected sum. If you step on a cell that you have **previously stepped in** – print "**Sadly the horse is doomed in JUMPS jumps';**" where JUMPS is the number of cells passed.

You start always at bottom-right cell (row-1, col-1).

Create a function in JavaScript that solves this problem.

Example:

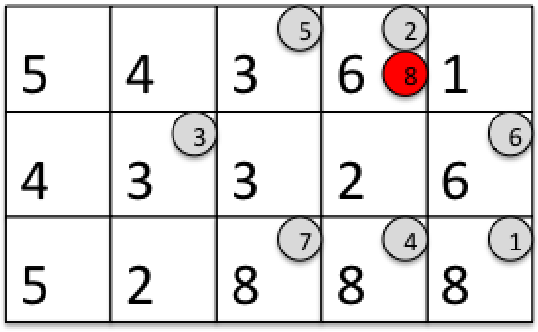
The **sum s 1,** the output should be: **"Go go Horsy! Collected 1 weeds"**



5	4	5	6	1
4	3	3	2	8
5	2	3	8	8

1	0	-1	-2	-3
2	1	0	-1	-2
4	3	2	1	0

Jumped on a **visited** cell, the output should be: **" Sadly the horse is doomed in 8 jumps "**



5	4	3	6	1
4	3	3	2	6
5	2	8	8	8

Input

The first value of the input arguments you will be the numbers R and C, separated by a space

- R is the number of rows in the matrix
- C is the number of columns in the matrix

The next R values in the input arguments will be exactly C directions

Output

Your function should return a single string:

- If **success**, return **"Go go Horsy! Collected SUM weeds"**, where sum is the sum of the visited cells
- If **fail**, return **"Sadly the horse is doomed in JUMPS jumps"**, where JUMPS are the number of cells from the path of the horse, including the visited cell.

Constraints

- **N will be between 2 and 30, inclusive**
- **C will be between 2 and 1000, inclusive**
- **Each**
- Allowed working time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Examples

Input	Output	Explanation

`args = [`
`  '3 5',`
`  '54561',`
`  '43328',`
`  '52388',`
`];`
	`Go go Horsy!`
	`Collected 1 weeds`
		`Explained in the example above`

Input	Output	Explanation

`args = [`
`  '3 5',`
`  '54361',`
`  '43326',`
`  '52888',`
`];`
	`Sadly the horse is`
	`doomed in 8 jumps`
		`Explained in the example above`