The Death Star

The Rebel Alliance has intercepted a high-frequency encrypted transmission from a hidden Imperial facility. The data contains the schematics for a new, modular **Death Star** hull plating system.

The hull is represented as a $(2N+1) \times (2M+1)$ grid. Each sector $C_{i,j}$ is either **reinforced plating** (white) or **exhaust vents** (black). Imperial engineering doctrine dictates that to maintain structural integrity, the reinforced plating must be **convex**: in any given row or column, the white cells must form a single, unbroken line, and all white cells are reachable from each other going only through white cells.

The Mission

Your droid, R2-D2, has successfully sliced into the central computer, but the connection is unstable. He cannot download the entire map at once. However, he can probe specific coordinates (x, y).

When R2-D2 probes a coordinate, the Imperial security system returns a "scrambled" parity check:

1. If the sector (x, y) is **reinforced plating** (white), the system returns the product of the total number of white cells in that row and the total number of black cells in that column.
2. If the sector (x, y) is an **exhaust vent** (black), the system detects the intrusion and returns -1.

The Rebel fleet knows that the central reactor core at (N, M) is always reinforced ($C_{N,M}$ is white).

Using no more than $(2N+1) \times (2M+1)$ probes, reconstruct the entire hull map so the Alliance can identify the weak points.

Implementation Details

You should implement the following procedure.

```
std::vector<std::vector<int>> compute_grid(int N, int M)
```

- N, M : positive integers representing the dimensions of the grid.
- This procedure returns a $(2N+1) \times (2M+1)$ grid G corresponding to the initial grid, such that $G[i][j] = 0$ if the cell $C_{i,j}$ is black, and 1 if it is white.

- This procedure is called exactly once per testcase.

This procedure can call the following procedure to ask a question :

```
int ask(int x, int y)
```

- x, y : two integers satisfying $0 \leq x < 2N + 1$, $0 \leq y < 2M + 1$
- This procedure returns the number of white cells on row x multiplied by the number of black cells on column y if the cell (x, y) is white, and -1 otherwise.
- This procedure can be called at most $(2N + 1) \times (2M + 1)$ times in a single execution.
- If your program gives (x, y) coordinates that are out of bounds, or if you call `ask` more than $(2N + 1) \times (2M + 1)$ times, your program will terminate and you will get a verdict of `WRONG ANSWER`.

Constraints

- $1 \leq N \leq 500$
- $1 \leq M \leq 1000$

Subtasks and Scoring

Let C be the number of calls to `ask` made in a testcase, we define :

$$X = \min\left(1, \frac{100e^{\frac{C-5716}{25000}} + 913}{500\left(e^{\frac{C-5716}{25000}} + 1\right)}\right)$$

Subtask	Score	Constraints
1	15	$N, M \leq 10$
2	$85X$	No further constraints

Your final score on a subtask is the minimum score over all testcases of the subtask. If your function returns a wrong output for at least one testcase of a subtask, your score for that subtask will be 0.

Example

Example Grid

```
5 6
00000000000000
00000000000000
00000000000000
00000110000000
00000110000000
00001111000000
00001100000000
00000000000000
00000000000000
00000000000000
00000000000000
```

Example queries

- Since the cell at coordinates $(5, 6)$ is a white cell, a call to `ask(5, 6)` will return the product of the number of white cells on row 5 and the number of black cells on column 6: There are 4 white cells on row 5, and 10 black cells in column 6, which means the procedure will return 40.
- A call to `ask(6, 6)` will return -1 as the cell at coordinates $(6, 6)$ is black.