Maze with Hint

This is a communication problem

There is a maze on an $N \times N$ board. Some cells are blocked and others are empty. All the cells on the edge of the board are blocked. You can move through the empty cells. At each moment, you can only see the state of the 4 neighboring cells, and you must decide which neighboring cell to move to. Your goal is to get from the bottom left corner to the top right corner of the maze in the least number of moves possible. It is guaranteed that these two cells are empty, and that there is a path of empty cells connecting them. Additionally, you will have the help of César. At the beginning of the game, César will give you an initial hint, which consists of a binary array. César will have access to the maze, and by observing it, he will generate a hint that you will receive at the start of the game.

Problem

Implement two functions. One that receives the maze information and generates a hint, and another function that, given the hint and only the information of the neighbors of the current cell, traverses the maze from the bottom left corner to the top right corner.

Implementation Details

Implement two functions. The first function, *Pista()*, receives a variable N, the size of the maze, and a vector of vectors, *Laberinto*, with the information of the blocked and empty cells. This function should return the hint as a vector of integers (which additionally have the restriction to be 1 or 0). The second function, *Juego()*, receives a vector of integers, the hint generated by the first program, and has access to the function $mover()$ to interact with the maze. When you call this function, you must specify a parameter that details the type of movement. The available parameters and processes are the following:

Parameter	Process
$mover(-1)$	The game starts.
$mover(0)$	You move from cell (x, y) to cell $(x, y+1)$.
$mover(1)$	You move from cell (x, y) to cell $(x+1, y)$.
$mover(2)$	You move from cell (x, y) to cell $(x, y-1)$.
$mover(3)$	You move from cell (x, y) to cell $(x-1, y)$.
$mover(4)$	The game ends.

Each time you call the move function, the process can be successful or fail. If you moved to an empty cell, or finished the game in the correct cell, the process is successful. In any other case, the process fails, the judge kills the program, and you receive 0 points for the case being processed.
Regardless of the parameter, the function will return an array of 4 booleans. Let's call it a. $a[i]$ is true if $mover(i)$ would be a valid move.

You must include the library *"Laberinto.h"* with the command *#include "Laberinto.h"* in order to be able to carry out the interaction.
The following is a template of the program you need to implement:

```
#include "Laberinto.h"
#include <bits/stdc++.h>
using namespace std;

vector<int> Pista(int N, vector<vector<int>> Laberinto) {
    vector<int> hint;
    // Implement this function.
    return hint;
}
void Juego(vector<int> hint) {
    // Implement this function.
}
```

The grader will run your program twice for each case.

Example

- The grader runs your program and calls the function

  $Pista(5, \{\{0, 0, 0, 0, 0\}, \{0, 1, 0, 1, 0\}, \{0, 1, 1, 1, 0\}, \{0, 1, 1, 1, 0\}, \{0, 0, 0, 0, 0\}\})$

  the maze is as follows:

0	0	0	0	0
0	1	0	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

- Your program would return a vector of integers, the hint. Let's say it returns the vector $\{0, 0, 0\}$.
- Then, the grader runs your program again and calls the function

  $$Juego(\{0, 0, 0\})$$

- A possible interaction would be the following:

Function called	returned variable
$mover(-1)$	{0, 1, 1, 0}
$mover(1)$	{0, 1, 1, 1}
$mover(1)$	{0, 0, 1, 1}
$mover(2)$	{1, 0, 1, 1}
$mover(2)$	{1, 0, 0, 0}
$mover(4)$	{1, 0, 0, 0}

- Given this sequence of operations, it is possible to show that in any subtask, you would obtain all the points in this test case.

Considerations

- $N \leq 200$.
- The 4 edges of the board consist of blocked cells.
- It is guaranteed that the bottom left and top right corner cells are empty, and that there is at least one path of empty cells between them.
- In the function *Pista()* must return a vector that consists of only 1's and 0's. Otherwise you will get 0 points.

Subtasks

Let's say the number of calls to the move() function is M, and the size of the hint is P. Also, let's call P_j the size of the hint of the model solution that the jury has for a specific maze.

- (5 points) There is no limit on M or P.
- (15 points) The value of M must be minimal for the given maze, and P must be at most P_j. It is guaranteed that there is no empty cell with four neighboring empty cells.
- (20 points) The value of M must be minimal for the given maze. There are no restrictions regarding P.
- (60 points)
  - To get the 60 points, the value of M must be minimal for the given maze and P must be at most P_j.

- Any solution with a P greater will be normalized between P_j and 2 times the optimal value of M. The points you will get for each case will be equal to

$$1 - \frac{P_j - P}{P_j - 2M}.$$

In any case, if your program has P greater than 2 times the optimal value of M, or if your second program does not manage to reach the top right corner of the maze, you will receive 0 points for that case.