Laberinto con Pista

Este es un problema de comunicación

Hay un laberinto en un tablero de tamaño $N \times N$. Algunas celdas están bloqueadas y otras están vacías. Todas las celdas del borde del tablero están bloqueadas. Te puedes mover por las celdas vacías. En cada momento, solo puedes ver el estado de las 4 celdas vecinas, y debes decidir a qué celda vecina moverte. Tu objetivo es llegar de la esquina inferior izquierda a la esquina superior derecha del laberinto, en la menor cantidad de movimientos posibles. Se garantiza que dichas dos celdas están vacías, y que existe un camino de celdas vacías que las une.
Además, tendrás la ayuda de César. Al inicio del juego, César te dará una pista inicial, que consiste de un arreglo binario. César tendrá acceso al laberinto, y a partir de observarlo, generará una pista, que recibirás al inicio del juego.

Problema

Implementa dos funciones. Una que reciba la información del laberinto y genere una pista, y otra función que dada la pista, y solamente la información de los vecinos de la celda actual, traverse el laberinto desde la esquina inferior izquierda hasta la esquina superior derecha.

Detalles de Implementación

Implementa dos funciones. La primera función, $Pista()$, recibe una variable N, el tamaño del laberinto, y un vector de vectores, $Laberinto$, con la información de las celdas bloqueadas y vacías. Esta función debe regresar la pista, como un vector de enteros (estos enteros deben ser 1 o 0). La segunda función, $Juego()$, recibe un vector de enteros, la pista que generó el primer programa, y tiene acceso a la función $mover()$ para interactuar con el laberinto. Cuando llamas dicha función, debes indicar un parámetro, que detalla el tipo de movimiento. Los parámetros y procesos disponibles son los siguientes:

Parámetro	**Proceso**
$mover(-1)$	Se inicia el juego.
$mover(0)$	Te mueves de la casilla (x, y) a la casilla $(x, y+1)$.
$mover(1)$	Te mueves de la casilla (x, y) a la casilla $(x+1, y)$.
$mover(2)$	Te mueves de la casilla (x, y) a la casilla $(x, y-1)$.
$mover(3)$	Te mueves de la casilla (x, y) a la casilla $(x-1, y)$.
$mover(4)$	Se acaba el juego.

Cada vez que llames la función mover, el proceso puede ser exitoso o fallar. Si te moviste a una casilla vacía, o finalizaste el juego en la casilla adecuada el proceso es exitoso. En cualquier otro caso, el proceso falla y el jurado mata el programa, y recibes 0 puntos en el caso que se esté procesando.

independientemente del parámetro, la función regresará un arreglo de 4 booleanos. Llamémosle a. $a[i]$ es verdadero si $mover(i)$ sería un movimiento válido.
Además, para llevar a cabo la interacción, debes incluir la librería *"Laberinto.h"* con el comando *#include "Laberinto.h"*.
Lo siguiente es un ejemplo de cómo se vería el programa que debes implementar:

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

vector<int> Pista(int N, vector<vector<int>> Laberinto) {
    vector<int> Pista;
    // Programa esta función.
    return pista;
}
void Juego(vector<int> Pista) {
    // Programa esta función.
}
```

El evaluador correrá tu programa dos veces por cada caso.

Ejemplo

- El evaluador corre tu programa, y llama la función

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

  el laberinto es el siguiente:

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

- Tu programa regresaría algún vector binario, la pista. Digamos que regresa el vector $\{0,0,0\}$.
- Entonces, el evaluador corre de nuevo tu programa, y llama la función

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

- Una interacción posible sería la siguiente:

Función llamada	variable que regresa
$mover(-1)$	$\{0, 1, 1, 0\}$
$mover(1)$	$\{0, 1, 1, 1\}$
$mover(1)$	$\{0, 0, 1, 1\}$
$mover(2)$	$\{1, 0, 1, 1\}$
$mover(2)$	$\{1, 0, 0, 0\}$
$mover(4)$	$\{1, 0, 0, 0\}$

- Dada esta secuencia de operaciones, es posible demostrar que en cualquier subtarea, obtendría todos los puntos que represente este caso.

Consideraciones

- $N \leq 200$.
- Los 4 bordes del tablero consisten de celdas bloqueadas.
- Se garantiza que las celdas de las esquinas inferior izquierda y superior derecha están vacías, y que existe al menos un camino de celdas vacías entre ellas.
- Si la función *Pista()* regresa un vector que no consista únicamente de 0s y 1s, recibirás 0 puntos.

Subtareas

Digamos que la cantidad de Llamadas a la función $mover()$ es M, y el tamaño de la pista P. También, llamémosle P_j al tamaño de la pista de la solución modelo que tiene el jurado para un laberinto específico.

- (5 puntos) No hay ningún límite sobre M o P.
- (15 puntos) El valor de M debe ser mínimo para el laberinto dado, y P debe ser a lo más P_j. Se garantiza que no existe una celda vacía con cuatro celdas vecinas vacías.
- (20 puntos) El valor de M debe ser mínimo para el laberinto dado. No hay restricciones respecto a P.
- (60 puntos)
  - Para obtener los 60 puntos, el valor de M debe ser mínimo para el laberinto dado y P debe ser a lo más P_j.

- Cualquier solución que tenga un P mayor, será normalizada entre P_j y 2 veces el valor óptimo de M, es decir, los puntos que obtendrás por cada caso serán igual a
$$1 - \frac{P_j - P}{P_j - 2M}.$$

En cualquier caso, si tu programa tiene P mayor que 2 veces el valor óptimo de M, o si tu segundo programa no logra llegar a la esquina superior derecha del laberinto, recibirá 0 puntos por ese caso.