Disclaimer: This problem is worth 200 points.

Charlie the Wondersweeper

Type	Input file	Output file	Time limit	Memory limit
Batch	stdin	stdout	2 seconds	128 MB

Statement

Charlie, once revered as the legendary Wonderhacker, has been laid off from Industrial Buzzword Management, as his unshakable habit of leaving all projects until the last minute was rightfully determined to be an unsustainable behaviour. Charlie has vowed to show his former company what he is truly capable of... that is after he satiates his newfound taste for some Windows XP nostalgia by becoming hopelessly addicted to minesweeper.

His only hope: to use his skills to write a bot so good at minesweeper that it exhausts all the joy of playing it. However, he can't muster the will to write the bot. Can you?

Minesweeper is a very simple game at heart: there is a grid of cells with W rows numbered from 1 to W, and H columns numbered from 1 to H that are filled with K hidden mines. You must uncover all empty cells while avoiding uncovering the cells with mines. When uncovering cell (x, y), if it contains a bomb, you lose and the game ends. If it is empty, the game continues and you will receive the number of mines in cells adjacent to (x, y). Cells are considered adjacent if they share either an edge or corner with (x, y).

Your program will also receive a starting location (X, Y) that that is surrounded by 0 mines, and is part of largest contiguous block of cells surrounded by 0 mines. Note that your program will not automatically uncover (X, Y) If this description of minesweeper isn't enough, have a go at playing the game yourself at `http://minesweeperonline.com/`.

The grader is not adversarial and there are no hand-made cases - in fact, the entire grid is determined before your code runs, and each test case was randomly generated according to the constraints.

Implementation

Anyone refusing to use C++ will be forced to conform. Deepest apologies to all you PHP users out there.

Do not print anything to stdout, or you will receive 0 points for the test case. You must not implement a main function. instead, you should include the line at the top of the file

```
#include "sweeper.h"
```

You must not implement a `main()` function, and instead you will be required to implement a single function,

```
void sweep_mines(int W, int H, int K, int X, int Y);
```

where W, H, K are the width, height, and number of mines on the board, and (X, Y) is a position with 0 mines surrounding it. This function will be called once at the start of the program and will be expected to call the following function provided by the grader:

```
int uncover(int x, int y);
```

on as many cells without mines as possible. Calling `uncover(x,y)` will attempt to uncover the cell at position (x, y). If the cell has a mine, or is outside the bounding box, or is the last cell to be uncovered, then the program will end and you will receive marks as described in the Scoring section. Otherwise, it will return a single integer, the number of bombs that surround it. Note that you still need to uncover cell (X, Y), it isn't automatically done for you.

Experimentation

In order to experiment with your code on your own machine, please download the provided files `sweeper.h` and `grader.cpp`, which should be placed in the same directory as your code. Compile your grader with the following command:

```
g++ grader.cpp sweeper.cpp -o sweeper -std=c++11 -O2 -static
```

This will create an executable `sweeper`, which you can then run. If you have trouble compiling, please send a message in the Communication section of CMS.

This grader will read in 4 integers on the first line, W, H, X and Y, W and H specifying the dimensions of the grid, and X and Y optionally specifying a starting location. If you wish for the grader to automatically select a starting location the same way the judge will, then simply leave X and Y to be zero.

Then it will read in H lines of W characters. The x-th character of the y-th line will be 'X' if the cell at position (x, y) is a mine, '.' if it doesn't contain a bomb. The grader will automatically calculate the number of bombs.

Sample Grader Input and Sample Session

```
10 10 0 0
..........
..........
..XX..XX..
..XX..XX..
....XX....
...XXXX...
...XXXX...
...X..X...
..........
..........
```

One possible interaction is shown below:

Grader	Student	Description
`sweep_mines(10,10,20,1,1)`		The grader initialises your program. Note that the grader automatically filled in values for (X, Y) as you specified $(0, 0)$ for (X, Y)
	`uncover(1,1)`	Note that you still have to call `uncover()` on your starting cell.
`uncover(1,1)` returns 0		There are 0 mines surrounding $(1, 1)$, Hence it returns 0.
	`uncover(2,2)`	Your program uncovers $(2, 2)$. Good choice, since it is neighbouring to $(1, 1)$ which has 0 neighbouring mines, neither can $(2, 2)$.
`uncover(2,2)` returns 1		There is one mine neighbouring $(2, 2)$. Hence it returns 1
	`uncover(3,3)`	Your program feels very adventurous and uncovers $(3, 3)$
Grader terminates from within `uncover(3,3)`		Oh no! The cell you uncovered contained a mine.

The sample grader will not score your program, instead it will simply output how many cells you cleared.

Scoring

As there are often cases where you are forced to guess the locations of the mines, you will be judged based on the number of cells you uncover before you hit a mine, query an invalid cell, or clear all the mine-free cells. Your score for each test case will be determined by two constants determined by the sub-task, hi and lo. Each test case will be out of 100 marks. Let the fraction of mines you clear be u

- If $u \geq hi$, then you will receive full marks for the test case (AC).
- If $u \leq lo$, then you will receive 0 marks
- If $lo < u < hi$ then your score will be linearly interpolated between 0 and 90. That is, your score will be

$$\frac{9(u - lo)}{10(hi - lo)}$$

Subtasks and Constraints

- $1 \leq X \leq W$
- $1 \leq Y \leq H$
- Within all subtasks, the values of W, H, K, lo and hi will be constant between each test case. These values are summarised in the following table:

Number	Points	W	H	K	lo	hi
1	20	10	10	10	0	1
2	40	16	16	40	0	1
3	80	30	16	99	0.5	1
4	60	200	200	8000	0.5	0.995

We struggled for a while getting CMS to score this problem the way we wanted it to: CMS regards each test case to represent one subtask.

In actual fact:

Cases	Subtask
1-5	1
6-15	2
16-35	3
36-50	4