Finding a path through a maze is a popular problem for computers. In this problem, a maze will consist of a rectangular array of square cells, each of which may have walls on the north, south, east and/or west sides of the cell. One cell will be identified as the starting point, and another will be identified as the goal. Your task is to find the unique route from the starting point to the goal, label each cell in the path with its sequence in the path, identify the cells that were visited but that were not in the path, and display the maze.

The algorithm you use to find a path through the maze must be the one described below. Imagine a robot is positioned in the starting cell. The robot first attempts to go west from that cell, then north, then east, then south, in sequence. The robot can move in the selected direction if

(a) there is no wall preventing it from moving in that direction, and

(b) it has not yet been in the next cell in that direction.

When the robot reaches the goal, its trip is over. If the robot reaches a cell at which no further progress is possible, it retreats to the previous cell it occupied and attempts to move in the next untried direction.

Consider the simple maze shown on the left below. It is two cells high and three cells wide. The starting cell is labeled 'S' and the goal cell is labeled 'G'. When the robot starts, it would first try to move west (left), but finds a wall. It then tries to move north (up), and is again blocked by a wall. A wall also prevents it from moving east (right), so it finally tries to move south (down), and succeeds. From the new cell it will eventually move east. Here it repeats its movement algorithm. Although no wall blocks its potential westward movement, it has already 'visited' the cell in that direction, so it next tries to move north, and is successful. Unfortunately, after moving north, it finds no way to extend its path, and so it retreats to the previously occupied cell. Now it tries to move east, and is successful. From that cell it will move north, and there it finds the goal. The maze that would be displayed on the output is shown on the right below. Note that the starting cell is labeled '1', each cell in the path to the goal (including the one containing the goal) is labeled with its sequence number, and each cell that was visited but is not in the path is labeled with question marks.

```
+---+---+---+        +---+---+---+
| S |   | G |        |  1|???|  5|
+   +   +   +        +   +   +   +
|           |        |  2   3   4|
+---+---+---+        +---+---+---+
```

Input

View the maze as an array of cells, with the northernmost row being row 1, and the westernmost column being column 1. In the maze above, the starting cell is row 1, column 1, and the goal cell is row 1, column 3.

There will be one or more mazes to process in the input. For each maze there will first appear six integers. The first two give the height (number of rows) and width (numer of columns) of the maze (in cells). The next two give the position (row and column number) of the starting cell, and the last two give the position of the goal. No maze will have more than 12 rows or 12 columns, and there will always be a path from the starting point to the goal.

Following the first six integers there will appear one integer for each cell, in row major order. The value of each integer indicates whether a cell has a wall on its eastern side (1) and whether it has a wall on its southern side (2). For example, a cell with no eastern or southern wall has a value of 0. A cell with only a southern wall has a value of 2. A cell with both an eastern and a southern wall has a value of 3. The cells on the periphery of the maze always have appropriate walls to prevent the robot from leaving the maze; these are not specified in the input data.

The last maze in the input data will be followed by six zeroes.

Output

For each maze, display the maze as shown in the example above and the expected output below, appropriately labeled and prefixed by the maze number. The mazes are numbered sequentially starting with 1. Print two blank lines after each dataset.

Sample Input

```
2 3 1 1 1 3
1 1 0
0 0 0

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

0 0 0 0 0 0
```

Sample Output

```
Maze 1

+---+---+---+
|  1|???|  5|
+   +   +   +
|  2   3   4|
+---+---+---+


Maze 2

+---+---+---+
|??? ???|???|
+   +---+   +
|  3   4   5|
+   +---+   +
|  2   1|  6|
+   +---+   +
|       |  7|
+---+---+---+
```