Round 5: The Jam Round

Problem 1 : Minesweeper

Minesweeper is a computer game that became popular in the 1980s, and is still included in some versions of the Microsoft Windows operating system. This problem has a similar idea, but it does not assume you have played Minesweeper.

In this problem, you are playing a game on a grid of identical cells. The content of each cell is initially hidden. There are M mines hidden in M different cells of the grid. No other cells contain mines. You may click on any cell to reveal it. If the revealed cell contains a mine, then the game is over, and you lose. Otherwise, the revealed cell will contain a digit between 0 and 8, inclusive, which corresponds to the number of neighboring cells that contain mines. Two cells are neighbors if they share a corner or an edge. Additionally, if the revealed cell contains a 0, then all of the neighbors of the revealed cell are automatically revealed as well, recursively. When all the cells that don't contain mines have been revealed, the game ends, and you win.

For example, an initial configuration of the board may look like this ('*' denotes a mine, and 'c' is the first clicked cell):

.....**.

....*.....

..c..*....

........*.

....

There are no mines adjacent to the clicked cell, so when it is revealed, it becomes a 0, and its 8 adjacent cells are revealed as well. This process continues, resulting in the following board:

.....**.

1112*.....

00012*....

00001111*.

00000001..

At this point, there are still unrevealed cells that do not contain mines (denoted by '.' characters), so the player has to click again in order to continue the game.

You want to win the game as quickly as possible. There is nothing quicker than winning in one click. Given the size of the board (R x C) and the number of hidden mines M, is it possible (however unlikely) to win in one click? You may choose where you click. If it is possible, then print any valid mine configuration and the coordinates of your click, following the specifications in the Output section. Otherwise, print "Impossible".

Input

The first line of the input gives the number of test cases, T. T lines follow. Each line contains three space-separated integers: R, C, and M.

Output

For each test case, output a line containing "Case #x:", where x is the test case number (starting from 1). On the following R lines, output the board configuration with C characters per line, using '.' to represent an empty cell, '*' to represent a cell that contains a mine, and 'c' to represent the clicked cell.

If there is no possible configuration, then instead of the grid, output a line with "Impossible" instead. If there are multiple possible configurations, output any one of them.

Limits

Memory limit: 1 GB.
$0 \le M < R * C$.

Small dataset

Time limit: 60 seconds.

$1 \le T \le 230$.

$1 \le R, C \le 5$.

Large dataset
Time limit: 120 seconds.

$1 \le T \le 140$.

$1 \le R, C \le 50$.

Sample Output

```
Case #1:
Impossible
Case #2:
Impossible
Case #3:
......*
.c....*
.......
..*....
Case #4:
**********
**********
**********
****....**
***.....**
***.c...**
***....***
**********
**********
**********
```

Sample Input

```
5
5 5 23
2 2 1
4 7 3
10 10 82
```

Problem 2 :Nautical Map

Given that a ship has sailed from a position and it has to travel to its destination. Due to the storm, the visibility is next to none but the cargo aboard the ship is required to be delivered to the destination on time the next day.

The captain has a map which highlights the obstacles which they have to navigate and/or avoid to reach the destination safely.

According to standard navigation rules, each obstacle has a radius of 100m and cannot travel diagonally due to the nature of the cargo.

The values given for the longitude and latitude need to be expanded to get the nearest position of the obstacle which can further be made precise by taking the nearest whole number after conversion.

Given you want to expand the values for longitude and latitude you will have to map it to the following coordinates of longitude and latitude: [11.95, 64.31]

Write an algorithm to draw a map of 100 across by 30 down (100x30) which shows the shortest route to reach the destination without the ship taking any damage.
The legend for the map,

1. "_" - should represent potential path or sea,
2. "*" - should represent obstacles in the given input
3. "0" - should represent the selected path width.
4. "1" - should represent the starting and ending location for the ship.

First line of input is number of ships
First value is longitude and second value is latitude

Input

O = Number of Obstacles
Ship Starting point , Ending point
Obstacles

Sample

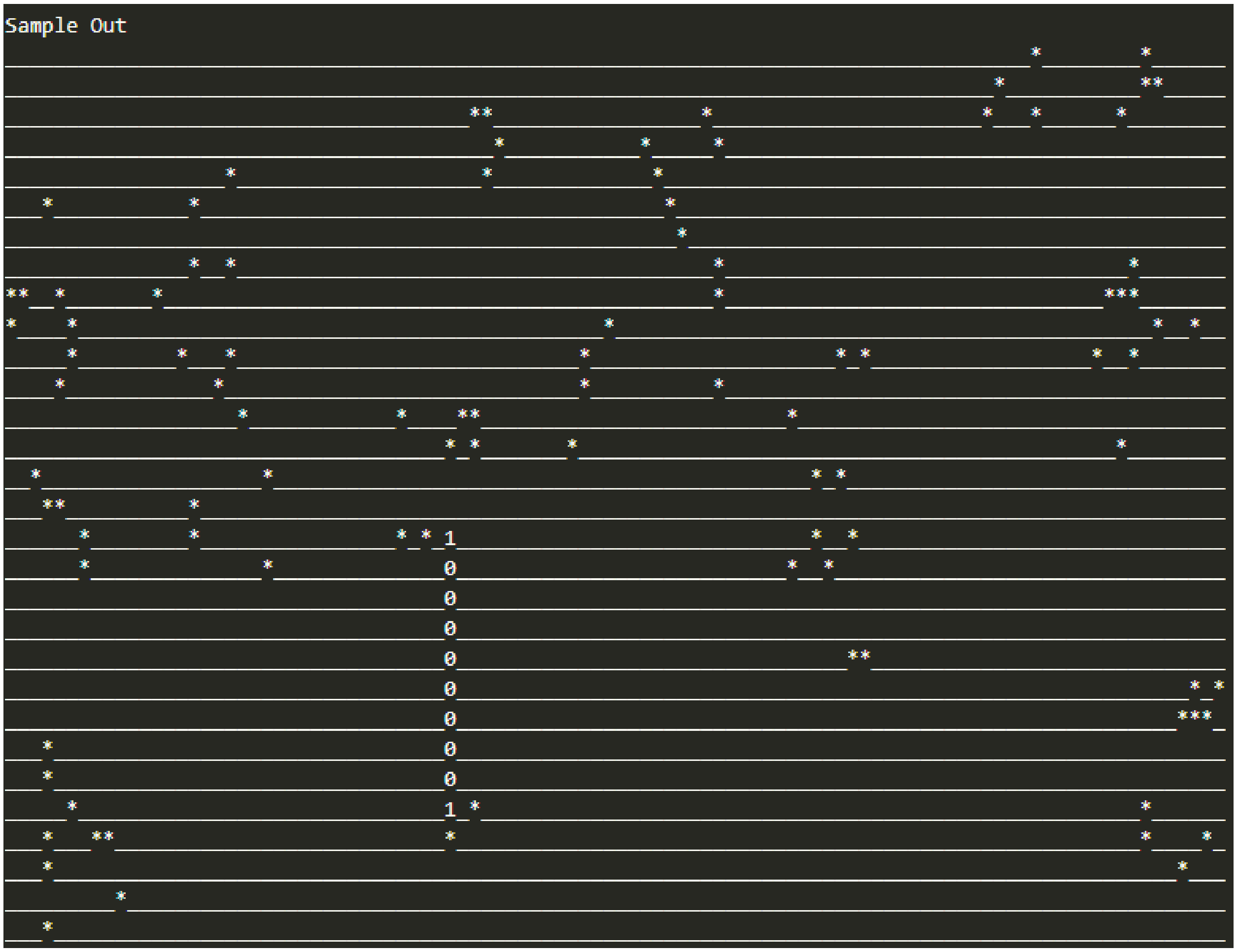
Input

```
98
3.013, 0.249, 3.013, 0.389
7.029, 0.000
7.782, 0.000
6.778, 0.016
7.782, 0.016
7.866, 0.016
3.180, 0.031
3.264, 0.031
4.770, 0.031
6.695, 0.031
7.029, 0.031
7.615, 0.031
3.347, 0.047
4.351, 0.047
4.854, 0.047
1.506, 0.062
3.264, 0.062
4.435, 0.062
0.251, 0.078
1.255, 0.078
4.519, 0.078
4.603, 0.093
1.255, 0.109
1.506, 0.109
4.854, 0.109
7.699, 0.109
0.000, 0.124
0.084, 0.124
0.335, 0.124
1.004, 0.124
4.854, 0.124
7.531, 0.124
```

7.615, 0.124
7.699, 0.124
0.000, 0.140
0.418, 0.140
4.100, 0.140
7.866, 0.140
8.117, 0.140
0.418, 0.155
1.172, 0.155
1.506, 0.155
3.933, 0.155
5.690, 0.155
5.858, 0.155
7.448, 0.155
7.699, 0.155
0.335, 0.171
1.423, 0.171
3.933, 0.171
4.854, 0.171
1.590, 0.187
2.678, 0.187
3.096, 0.187
3.180, 0.187
5.356, 0.187
3.013, 0.202
3.180, 0.202
3.849, 0.202
7.615, 0.202
0.167, 0.218
1.757, 0.218
5.523, 0.218
5.690, 0.218
0.251, 0.233
0.335, 0.233
1.255, 0.233
0.502, 0.249
1.255, 0.249
2.678, 0.249
2.845, 0.249
5.523, 0.249
5.774, 0.249
0.502, 0.264

1.757, 0.264
5.356, 0.264
5.607, 0.264
5.774, 0.311
5.858, 0.311
8.117, 0.327
8.285, 0.327
8.033, 0.342
8.117, 0.342
8.201, 0.342
0.251, 0.358
0.251, 0.373
0.418, 0.389
3.180, 0.389
7.782, 0.389
0.251, 0.404
0.586, 0.404
0.669, 0.404
3.013, 0.404
7.782, 0.404
8.201, 0.404
0.251, 0.420
8.033, 0.420
0.753, 0.435
0.251, 0.451

Output



Problem 3 : Tea-Point Seating Capacity

Given that Shubham is opening a new branch for Tea-Point. He is trying to own and rent the ideal amount of furniture so that he can cater to the ever changing amount of people visiting your shop.

New furniture he wants to buy and rent includes benches and stools.

He wants to serve his customers from 06 in the morning till 12 midnight and will have customers changing at every 2 hour intervals.

He can rent or purchase in consecutive order only. If he's rented/purchased 1 bench then the next time he has to rent/purchase a pair of stools. But if he wants to purchase a stool individually, he can.

Write an algorithm to find the optimum seating capacity you should own the furniture for and the price for renting the extra furniture.

Input

First line of input T is number of test cases
First line of each test case has three integers B,C,D
B is per number of hours the renting is available
C is per unit rent price for stool with respect to B
D is per unit rent price for bench with respect to B
Second line of test case E determines number of customer at each time interval

Output

Output contains two space separated values. first is number sitting arrangement he should own and second is total amount he spent on renting

Case #1: [Owned seating capacity] [Total renting expense]

Limits

1<T<100 (small)
1<T<1000(large)
1<B<8

Sample input

```
2
1 10 20
10 10 5 15 25 30 45 85 85
4 10 20
1 2 3 4 5 8 8 8 9
```

Sample output

```
Case #1 : 25 2320
Case #2 : 5 50
```