Problem L. Mine Map

After the recent theft of the problemset for the ACM ICPC World finals (by notorious british super spy James B.—we reported on this some weeks ago), the ACM has decided to store all future problemsets in a high security building. The security board endowed with the job of creating this new vault had the brilliant idea to build it in the form of a giant maze. Essentially, this maze consists of a bunch of square rooms, arranged in the form of a square matrix, with all the rooms connected to each other by a series of doors. Going through them is the only way to get to the center where the problemsets are stored.
Obviously, it is not that hard to get through a maze in which all room are connected to each other. So, to make things more dangerous for would-be intruders, some of the rooms are booby trapped with mines. If somebody enters the central room of the vault containing the problemsets, these mines are activated. Afterwards, opening a door leading to a room with a mine in it will trigger an alarm, and all security doors close immediately, trapping the intruder. This way, the ACM can find out who sent the spy and disqualify all teams of that nation.
But recently the security board became aware of a new scanning device able to detect the mines, once they are activated. This detector could be used from within any of the rooms of the vault, and would be able to tell the user whether any of the up to eight adjacent rooms contains a mine, or not. Unfortunately, the board already ordered a batch of these mines, and now doesn't want to have to admit that this might have been a mistake. Instead, they simply want to spread the mines in such a way that it is difficult to leave the center by just using the device. Your are assigned to the team building the vault in order to help them evaluate their designs.

Problem

A vault has the form of a quadrangle, with sides that have odd length. Each room in the vault can be described by a pair of coordinates, indicating that horizontal and vertical offset relative to a fixed corner of the building. In each room, there are doors leading to the neighboring rooms; more importantly, the mine detector can detect mines in all of the up to eight adjacent rooms. The device can only tell you whether there are any mines nearby, but not how many there are.
Your job is to create a special map from each of the vault design drafts. On the map, mark all rooms somebody starting from the center room (which is guaranteed to not contain a mine) could safely reach with the help of the new detector and the following simple strategy: When you are in a room where the detector reports no adjacent mines, search all surrounding rooms. Otherwise, do not risk triggering a mine and do not advance farther from this room (you might reach one of the surrounding rooms via another "safe" route later on, though).
Thus, if the intruder is in a room not next to any mines, he will be able to go to all surrounding rooms—mark such a room with a "`.`". If the intruder enters a room which is next to one or more mines, he will retreat—mark these rooms with a "#". To be able to verify your work, the security board also wants you to mark the position of each mine with a "`*`". Finally, all remaining rooms should be marked with a "`?`".

Input

The first line contains the number of scenarios. Each scenario starts with a line containing the odd integer n ($1 < n < 300$) of the vault, indicating the length of one of its outer walls. This is followed by the number m of mines (which is positive and only limited by the number of rooms in the vault).
Next comes m lines, each containing two integers r and c ($1 \leq r, c \leq n$), which give the row and the column of a mine.

Output

The output for every scenario begins with a line containing "`Scenario #i:`", where `i` is the number of the scenario starting at 1. Then print an ASCII representation of the map of the vault as described above. Terminate the output for the scenario with a blank line.

Sample Input

```
3
3
1
1 2
5
4
3 1
1 3
3 5
5 3
5
2
1 1
5 3
```

Sample Output

```
Scenario #1:
?*?
?#?
???

Scenario #2:
??*??
?###?
*#.#*
?###?
??*??

Scenario #3:
*#...
##...
.....
.###.
.#*#.
```