Chess Variants Showdown

Simon the Chess Player is excellent at playing chess, a true chessmaster. He's not only great at traditional chess, but also its different variants. Chinese chess, Japanese chess, hexagonal chess – it doesn't matter, he never lost a game in his life. Unfortunately it means that no one really likes to play with him, so he has to somehow entertain himself.

This time he came up with a following puzzle. There is a generalized chess board, with **n** rows and **n** columns. Rows are numbered starting with 1, top to bottom. Columns a numbered starting with 1, left to right. There are kings on some of the squares. You have to place some of your pieces on the empty squares, such that you check every king. (King is said to be in check, if given current state of the board there exists a move that would capture the king – we always assume it's our turn, so movement of the kings does not matter at all.) You can use the following pieces, from different chess variants, with different costs:

1. **bishop** (chess) – can move any number of squares diagonally (in four directions). Its movement is stopped by pieces that are already on the board (so it can only capture the first obstacle on his way).
2. **rook** (chess) – can move any number of squares horizontally or vertically, otherwise analogous to a bishop.
3. **knight** (chess) – can make two squares horizontally and one vertically, or two squares vertically and one horizontally, an L-shaped move (8 possibilities). It ignores any pieces in its way.
4. **golden general** (shogi) – can move into any of 8 neighboring squares, with two exceptions: it can't move diagonally backwards to the left or to the right ("backwards" means in the direction o increasing row numbers).
5. **silver general** (shogi) – can move into any of 8 neighboring squares, with three exceptions: it can't move directly to the left (the direction of decreasing columns), directly to the right or directly backwards.
6. **horse** (shogi) – can move just like bishop, and additionally it can move into any of 8 neighboring squares.
7. **phoenix** (dai shogi) – can move one square horizontally or vertically (4 possibilities), or two squares diagonally (4 possibilities), 8 possibilities in total. Like knight, it ignores pieces standing in its way.
8. **cannon** (xiangqi) – can move just like rook, but in order to capture a piece, there must be exactly one other figure on the way between the cannon and the captured piece (doesn't matter to which player it belongs). Cannon jumps over that figure without capturing it.

When in doubt, please consult movement rules from the original games (they are unchanged). Caution: horse from shogi is a promoted bishop, and NOT a promoted rook (*ryūma*, not *ryūō*).

Please help Simon with his arbitrary problem – find an arrangement of pieces that checks all of the kings, with the minimum possible sum of costs.

Input

The first line contains a single integer **t**, denoting number of testcases. Then, testcases follow.

First line of the testcase contains one integer **n** ($2 \leq \mathbf{n} \leq 50$) – board size. Then, **n** lines follow, each containing **n** characters. j–th character in the i–th line corresponds to chessboard square from the i–th row and j–th column – if it's #, the square contains a king. If it's _, square is empty. For every square with a king, at least one of the 8 neighboring squares is empty.

In the last line of the testcase there are 8 integers $\mathbf{x}_i$ ($1 \leq \mathbf{x}_i \leq 10^6$) – figure costs, in the order like in the problem description.

Output

For every testcase you should output two integers: **f** and **g** ($1 \leq \mathbf{f} \leq \mathbf{n}^2, 1 \leq \mathbf{g}$) – number of pieces in your solution, and the total cost. Then you should produce a description of **f** pieces.

Description of one piece consists of three integers **x**, **w**, **k** ($1 \leq \mathbf{x} \leq 8, 1 \leq \mathbf{w}, \mathbf{k} \leq \mathbf{n}$) – type of piece (number as in the problem description), row number and column number. The arrangement of pieces should be made so every king is in check by at least one of your pieces. You can place your chess pieces only on empty squares, and you can place at most one piece on a square.

Example

Input	Output

1
5
___##
__#__
##_#_
_##__
##___
1 1 1 1 1 1 2 1
	3 4
	2 1 1
	7 3 3
	4 5 3

Explanation

A rook in the upper left, phoenix right in the middle and a golden general in the middle of the last row will check every king. Phoenix costs 2 and the rest costs 1, so total score is $1 + 2 + 1 = 4$.

Scoring

Score for one testcase is **g** – sum of costs for all pieces. Total score is a sum of all **g**'s in a file.

This is a minimization task – the less points, the better.