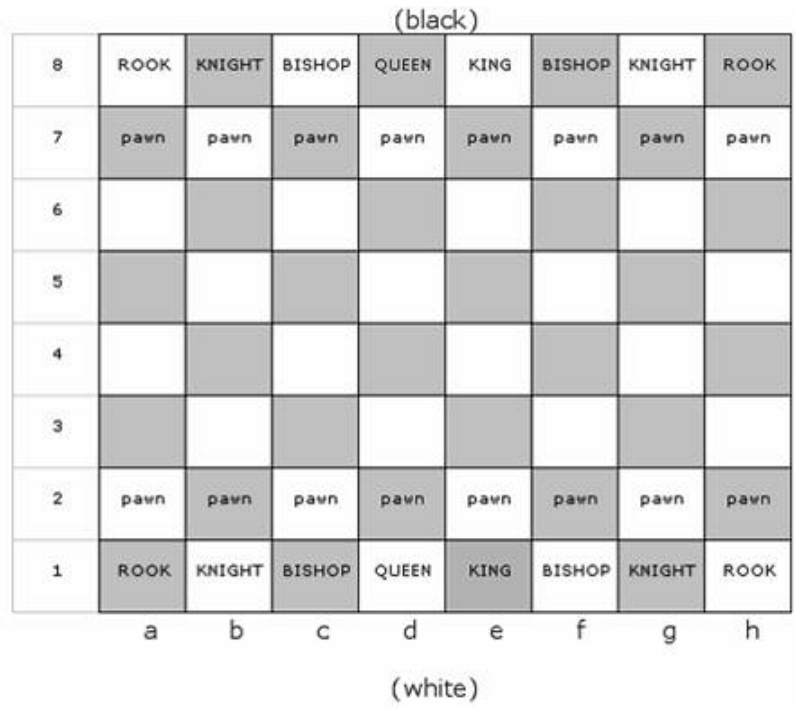
A difficult endgame in chess which even Grand Masters have difficulties mastering is queen + king versus rook + king. You are to write a program which calculates how the game will end assuming both sides play "perfectly". If the result is that one of the players will win, you should also print the shortest number of moves the winning side has to make (assuming perfect play by the opponent) before he mates. Finally, you should calculate which moves are the best for the player to move. That is, if the player to move has mate in 4, you should output all moves that will lead to mate in 4. Conversely, if the moving side will lose in 7 moves no matter how he plays, you should output the moves leading to this end, and not the moves which will cause an even quicker loss. In case perfect play will lead to a draw, output all moves which will lead to a draw.



Picture: A Chess Board at the start of a conventional game.

Input

The first line in the input file contains an integer n ($0 < n \leq 1000$) which is the number of chess positions to follow.

Next follows n lines, each describing one chess position. The line will begin with the description of the chess pieces on the board, which will always be four — two kings, a rook and a queen. A chess piece will be described as `WKa4` (White King at a4), `BQh3` (Black Queen at h3), `WRf7` (White Rook at f7). Capital letters will be used to describe the piece (`K`, `Q`, `R`) and the color (`W`, `B`), and `[a-h][1-8]` to describe the square. Each piece will be separated with at least one blank. Last on the line will be the side to move, either '`W`' (white) or '`B`' (black).

You may assume that the position is legal (no two pieces standing on the same square), that the rook and queen will have different colors, and that the player which is not to move will not be checkmate (however, the opposite might be true). You may also assume that castling will not be allowed anytime during the game for the player with king + rook.

Output

For each test case you should output one of the following:

- `White mates in` x
- `Black mates in` x
- `Draw`

where x is the minimal number of moves the winning side has to make to win. For instance, if white is to move but he is checkmate, you should output '`Black mates in 0`'

Following this you should, on a line by itself, output the best moves by the side to move. A move is written by first expressing which piece to move (`K`, `Q`, `R`), an optional 'x' if the move is a capture, the destination square and finally an optional character '`+`' (if the move results in a check) or '`#`' (if the move delivers mate). The list should be sorted in lexicographical order, see sample output.

In case the moving side is checkmate, output an empty line.

Output a blank line after each testcase.

Sample Input

```
6
WKa1 BKa3 WRc2 BQd2 B
BRf5 BKg5 WKb4 WQc5 W
WQb1 BRb3 WKd1 BKd3 B
WKd1 BKf1 BRg1 WQf3 B
WQb6 WKc6 BKf6 BRg6 W
WQb6 WKc6 BKf6 BRg6 B
```

Sample Output

```
Black mates in 2
Qd1+ Qe1+

White mates in 27
Qc7

Black mates in 1
Rxb1#

White mates in 0


White mates in 23
Kd5+ Qe3

Draw
Ke5+ Ke7+ Kf5+ Kf7+ Kg5+ Kg7+
```