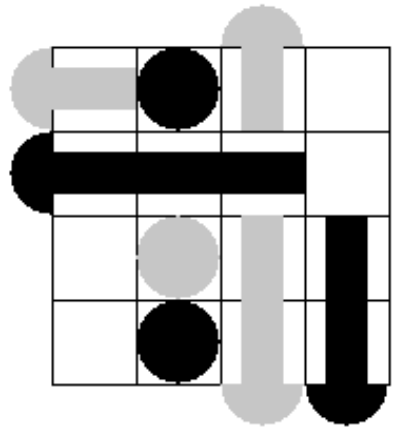
The Game of Euler is played between two players on a 4×4 board. The board is initially empty. The players alternatively put pins of different lengths on the board. You can either put them from one of the sides, in which case the pin will cover the same number of squares as the length of the pin (1, 2 or 3), or you can put it perpendicular to the board (pushing it straight down), covering exactly 1 square. A pin may only cover squares that are not covered already. The player who puts the last pin, and thus makes all 16 squares covered, loses. Both players have an infinite supply of pins of length 1, 2 and 3.

Consider the position in the picture to the right. If the player to move covers one of the two squares in either corner, the opponent will cover both squares in the opposite corner, so the first player will have only one move and will thus lose. But if the first player covers both squares in one corner, the opponent will cover only one square in the other corner, winning again. So the first player will lose the game no matter what move he makes. We say that such a position is *losing* for the player to move, because no matter which move he makes, he will lose the game if the opponent plays "perfectly" (that is, make the best moves). If the position is not losing, it is *winning*. Since fewer and fewer squares remains uncovered as the play progresses, the game will always end with a loser and a winner (never a draw).



Input

The first line in the input contains an integer N the number of test cases to follow ($N < 100,000$). Each test case contains of 4 lines, each line containing four character. These lines represent which squares of the board have been covered so far. A covered square is indicated by a '`X`', an uncovered square is indicated by a '`.`'. At least one square on the board will be uncovered. Each test case is preceded by a blank line.

Output

For each test case output a single line containing either '`LOSING`' or '`WINNING`' depending on whether the position is losing or winning for the player to move.

Sample Input

```
3

XXX.
XXX.
.XXX
.XXX

XXXX
...X
XX.X
XX.X

....
....
....
....
```

Sample Output

```
LOSING
WINNING
LOSING
```