Problem J. Tetra-puzzle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Tetra-puzzle is a turn-based game for one player resembling Tetris. The game goes on a square board of size 5×5 squares. Initially, the board is empty.

On each turn, the player has to place a tetramino of the given kind on the board. A tetramino is a side-connected piece consisting of four squares. Each piece can be rotated, flipped and translated, and has to be placed so that it occupies four empty squares of the board. After that, take all rows and columns of the board where all squares are filled, and simultaneously clear them up: every square of these rows and columns becomes empty again. The player loses if they can not make a move.

There are five kinds of tetramino pieces in total. Each kind is denoted by its name, an uppercase English letter resembling the form of the piece:

```
.....   .....   .....   .....   .....
.*...   .*...   .**..   .***.   .**..
.*...   .*...   .**..   ..*..   ..**.
.*...   .**..   .....   .....   .....
.*...   .....   .....   .....   .....

  I       L       O       T       Z
```

The game has two modes: basic mode and preparation mode. In basic mode, the player gets n pieces one by one, and has to place the given piece before getting the piece for the next move. The goal is to make all n moves successfully.

In preparation mode, the player plans their next game in basic mode. For preparation, the player gets n randomly generated pairs of pieces right away: the pairs for first, second, ..., n-th move. From each pair, the player has to select one piece, and that piece is what the player will get for the respective move in basic mode.

Your task is to successfully complete the game, first in preparation mode and then in basic mode using the prepared pieces.

Interaction Protocol

In this problem, your solution will be run twice on each test. Each line of input is terminated by an end-of-line character.

During the second run, this is an interactive problem. Do not forget to flush the output immediately after printing each move!

First Run

During the first run, you play in preparation mode. The first line contains the word "`prepare`". The second line contains an integer n, the number of turns ($1 \le n \le 1000$). The third line contains n space-separated pairs of pieces: two choices for first, second, ..., n-th move. Each pair is given by two uppercase letters from the set "`ILOTZ`" corresponding to the kinds of pieces in the pair. The letters in a pair are different and can be given in any order. In each test, each pair is selected randomly in advance, equiprobably from all possible choices and independently from other pairs.

Print a line containing n letters without spaces: for each move, print which one of the two pieces you choose.

Second Run

During the second run, you play in basic mode. The first line contains the word "`play`". The second line

contains an integer n, the number of turns which is the same as during the first run ($1 \le n \le 1000$). Then follow n turns.

On each turn, firstly, the solution reads a line with a single letter: the kind of tetramino selected for this turn during the first run. In response, print the board with the given piece already placed on it, but before clearing up the filled rows and columns. The board takes of 5 lines each of which consists of 5 characters. Character "`.`" (dot, ASCII code 46) means an empty square, character "`#`" (hash, ASCII code 35) is a square occupied on one of the previous turns, and character "`*`" (asterisk, ASCII code 42) is a square of the latest piece put on the board.

If the above formatting rules are satisfied, but the move is invalid, the solution will get "`Wrong Answer`" for the test.

If the move is valid, all filled rows and columns are cleared up, and the occupied squares must be denoted by "`#`" character further on. After that comes the next turn.

If all n turns already happened, the solution will get "`OK`" for the test.

Example

For each test, the input during the second run depends on the solution's output during the first run.

Below we show two runs of a certain solution on the first test. Empty lines are added only for readers' convenience: there will be no empty lines during a real run.

standard input	*standard output*
prepare 6 TO LO ZI LI LT LT	OLZITT

standard input	*standard output*
play 6 O	..... **... **... .... L ..*** ##..* ##... .... Z ..### ##**# ##.** .... I ..### ..*.. ##*## ..*.. ..*.. T ***## .*... .... T ..... *#... **... *.... .....