Timeweaver

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

Long ago, an **ancient civilization** flourished on an island in the heart of the sea. They revered the light as sacred, and left their traces—written in a language now forgotten.

But time moved on. The people vanished. The light fell silent. What remains are scattered fragments—records faded, cracked, and worn by the centuries.

After many long years, an explorer named Brue set foot on the forgotten island. The land lay silent—but fragments of light, scattered through the land, were whispering echoes of a past long buried.

Yet the records within the light had been warped, shattered, and twisted by time. Brue couldn't understand the language, and no clues remained to guide him. These records will never be deciphered, unless...

Timeweavers,

You were born from the fractures of time. From scattered fragments, you rethread what was lost, and let forgotten pasts stir once more in the present.

The records of the past are already distorted, and the explorer who follows cannot read them. But if your hands reach what remains, the scattered language of light may find shape again—and the lost records, their form. Remember—the thread you weave may yet alter the fate of the island.

Method of Recording

A record is created using the following method. First, prepare a 10×10 grid. Each of the 100 cells is inscribed with either 0 or 1. This is called the **base grid**. The cell at the r-th row from the top and c-th column from the left is denoted as (r, c). An example base grid is shown below:

0	1	1	0	0	1	0	0	1	1
1	0	1	1	1	0	1	1	1	1
0	0	0	1	0	0	0	0	1	0
1	1	0	1	1	1	1	0	0	0
0	0	0	1	1	0	1	0	1	0
1	0	1	0	0	1	1	1	0	0
0	1	0	1	1	1	0	1	0	0
0	1	0	1	0	0	0	0	1	0
0	1	1	1	1	0	1	0	1	1
1	0	0	1	0	0	0	1	0	1

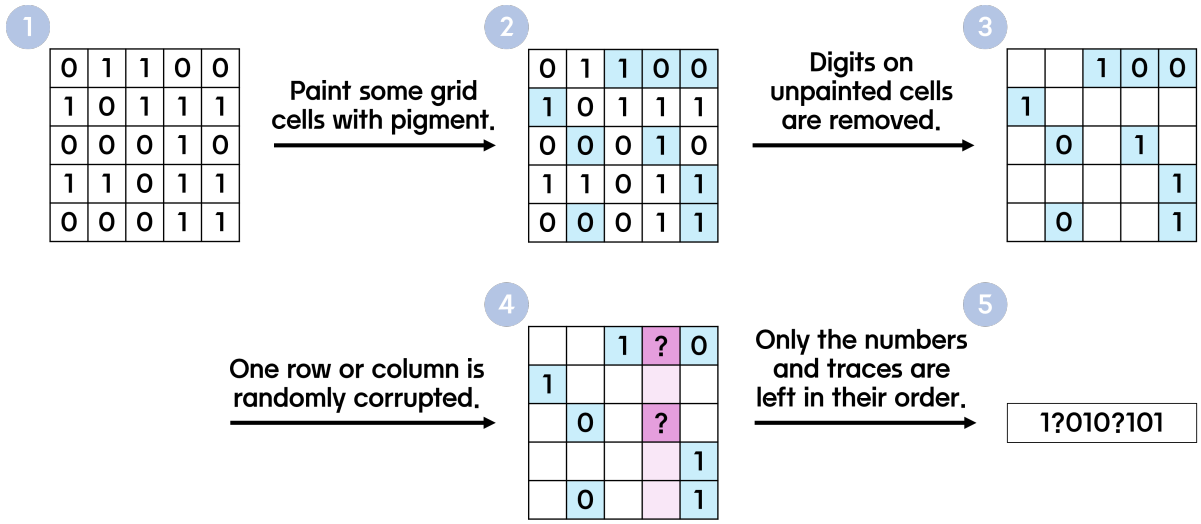
Once the base grid is prepared, a message can be recorded by painting some cells with pigment. The digits in painted cells are preserved over time, but the digits in unpainted cells are lost as the centuries pass.

Record Degradation

However, during the hundreds of years before discovery, the grid became distorted. The numbers in unpainted cells all disappear. Additionally, one entire row or column is randomly corrupted, making its numbers unreadable—though any paint marks on those cells remain visible.

Over time, the grid fades further, leaving only the numbers in painted cells (or traces of paint where numbers once were) and their relative order. The order is defined by ascending row number; if rows are equal, then by ascending column number.

The following illustration demonstrates this degradation process. The actual grid is 10×10, but a 5×5 example is used here for simplicity:



Deliver a method for recording and decoding these ancient messages and fulfill your mission.

Interaction Protocol

In this problem, your program will be run twice for each test. To validate your strategy, you will be recording and decoding various binary strings. The procedure can be summarized as follows:

- In the first run, you act as the recorder. You must record the given T binary strings into a grid.
- Next, the grader takes the grid you provided and transforms it according to the method described in the statement, producing resulting strings.
- In the second run, you act as the decoder. You must reconstruct the original binary strings from the resulting strings generated by the grader.

The first line contains a string R indicating your role. If R is `registrar`, you will act as the recorder. If R is `brue`, you will act as the decoder (Brue).

Based on your role, perform the corresponding recording or decoding procedure. **Flush the output buffer after every output.**

Role: Registrar

On the first line, output a single integer L—the length of the binary strings you can record and decode with your strategy.

Then, print ten lines representing the base grid. The i-th line should contain ten digits $G_{i1}, G_{i2}, \cdots, G_{i10}$, without spaces. This means that you inscribe digit G_{ij} in grid cell (i, j).

After that, read an integer T—the number of binary strings to record.

For each of the T iterations, repeat the following procedure.

- First, read the i-th binary string B_i of length L.
- Next, output ten lines indicating which cells you will paint. Use `'X'` to mark a painted cell and `'.'` for unpainted cells. **At least one cell should be painted.**

After printing the value of L, the base grid, and each painting pattern, you must flush the output buffer.

Role: Decoder

First, read an integer T—the number of messages to decode. Then, repeat the following T times.

- Read a string S_i consisting of the characters `'0'`, `'1'`, and `'?'`, representing the final, degraded state of the grid. Here, `'?'` indicates that the cell contains traces of paint. The order of the S_i strings given is the same as the first run.
- After the input, output the decoded binary string for S_i.

You must flush the output buffer after each output.

Input Constraints

- $1 \leq T \leq 1\,000$
- $B_{ij} \in \{\text{'0'}, \text{'1'}\}$ $(1 \leq i \leq T, 1 \leq j \leq L)$
- $S_{ij} \in \{\text{'0'}, \text{'1'}, \text{'?'}\}$ $(1 \leq i \leq T)$

Output Constraints

- $1 \leq L \leq 70$
- At least one of the 100 cells in the grid must be painted.

The example I/O is formatted with deliberate spacing for clarity. In actual interaction, do not print blank lines.

Scoring

You will receive no points if you fail to decode every record correctly.

If all records are successfully decoded, you will be awarded a score based on your value of $L = 10a + b$ $(0 \leq a,\ 1 \leq b \leq 10)$, according to the table below:

b	1	2	3	4	5	6	7	8	9	10
$a = 0$	2	3	4	5	6	8	9	10	11	12
$a = 1$	13	14	15	16	17	18	19	20	21	22
$a = 2$	23	24	25	26	27	28	29	30	31	33
$a = 3$	34	35	36	37	39	40	41	42	43	45
$a = 4$	46	47	49	50	51	53	54	56	58	60
$a = 5$	62	64	66	68	70	72	75	78	81	84
$a = 6$	89	100								

Examples

standard input	standard output
registrar 2 100101 000000	 6 1111100000 0000011111 1111100000 0000011111 1111100000 0000011111 1111100000 0000011111 1111100000 0000011111 X......... .X........ ..X....... ...X...... ....X..... .....X.... ......X... ....X.. ....X. ....X ....XX.... ....XX.... ....
brue 2 10101?0101 1001	 100101 000000

Note

After every output, you must flush the output buffer. If you don't, you may receive unexpected results. Use the following methods to flush:

- C: `fflush(stdout)`
- C++: `std::cout << std::flush`

- Java: `System.out.flush()`
- Python: `sys.stdout.flush()`

Refer to the language documentation for other languages.