Rook Detection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

This is an interactive problem.

Consider an $n \times n$ chessboard, where its rows are numbered 1 to n from top to bottom, and its columns are numbered 1 to n from left to right. The square on the x-th row and the y-th column is denoted as (x, y) and is either empty or occupied by a neutral rook. According to the rules of chess, a rook can move any number of squares along a row or column in one step but cannot leap over other pieces. In this scenario, the *control area* of a rook consists of the square it occupies and the squares it can reach in one step.

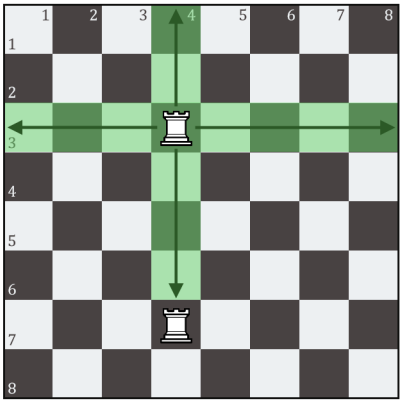

Figure 1: The control area of the rook at (3,4)

A square is considered "*controlled*" if and only if it belongs to one or more control areas. The known information is that all squares are controlled initially, so clearly, there are at least n rooks on the chessboard. Whether each square is empty or occupied by a rook, however, is unknown to you. Your task is to locate at least n of the hidden rooks by making no more than $\lceil \log_2 n \rceil + 2$ queries.

In each query, you need to split the chessboard into two tiers by determining which squares should comprise the upper tier and which the lower tier. The rooks are only allowed to move within the tier to which they belong, and they can leap over gaps but cannot leap over other pieces in the same tier. The definitions of control areas and controlled squares do not change. As a response, the interactor will inform you whether each square is controlled after the splitting. Then, the chessboard will return to the initial state.

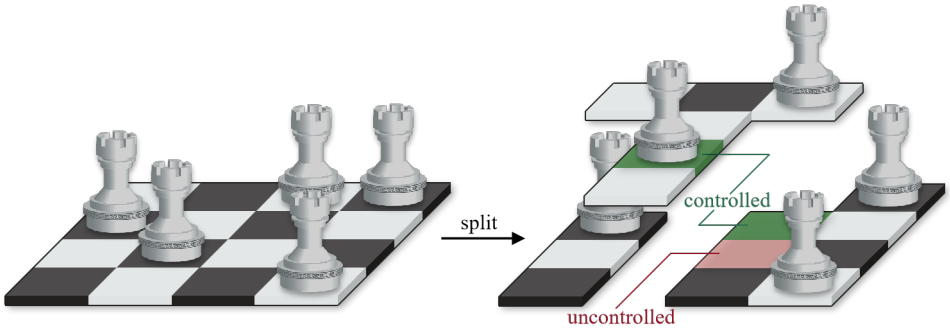

Figure 2: Split the T-shaped upper tier out of the chessboard

You may assume that the locations of all the rooks are fixed in advance. Note again that it is not necessary to find all the rooks when the actual number of rooks is greater than n.

Interaction Protocol

The first line contains an integer T ($1 \leq T \leq 10^4$), denoting the number of test cases.

The first line of each test case contains an integer n ($3 \le n \le 500$), denoting the size of the chessboard. It is guaranteed that the number of test cases that $n \ge 10$ does not exceed 5.

To initiate an interaction behavior, you should firstly print a character *opt* (*opt* $\in \{$ '?' , '!' $\}$) in one line and then print an $n \times n$ binary matrix (in n lines, each containing a binary string). After printing the binary matrix, do not forget to flush the output. To do this, use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, or `stdout.flush()` in Python.

opt = '?' represents making a query. If the character in the x-th row and y-th column of your binary matrix is '1', the square (x, y) will comprise the upper tier, or otherwise, it will comprise the lower tier. The interactor will return an integer *code* (*code* $\in \{0, 1\}$) in one line. If *code* = 1, you will get `Wrong Answer` because you made more than $\lceil \log_2 n \rceil + 2$ queries. If *code* = 0, you will then receive another $n \times n$ binary matrix from the interactor. If the character in the x-th row and y-th column of the received binary matrix is '1', the square (x, y) is controlled after the splitting, or otherwise, it is uncontrolled.

opt = '!' represents answering the locations. If the character in the x-th row and y-th column of your binary matrix is '1', the square (x, y) is occupied by a rook in your judgment, or otherwise, whether it is empty or occupied by a rook will not affect the verification of your answer. The interactor will return an integer *code* (*code* $\in \{0, 1\}$) in one line. If *code* = 1, you will get `Wrong Answer` because your answer failed to match the actual locations of the rooks or you did not find enough rooks. If *code* = 0, you will enter the next test case if the current test case is not the last one.

If you attempt to print *opt* that is neither '?' nor '!', or if your binary matrix is badly formatted, such as some of the n rows has a length other than n or contains a character that is neither '0' nor '1', the interactor will return *code* = -1 and you will get `Wrong Answer` as well.

Please terminate your program immediately when you receive some *code* other than 0 to avoid any unexpected verdicts.

Example

standard input	standard output
1 3 0 110 111 111 0 111 111 101 0	 ? 101 000 101 ? 111 100 100 ! 010 001 100

Note

The blank lines in the sample case are added for readability. In your output, extra spaces or blank lines will be ignored.