Problem H
QC QC
Time limit: 10 seconds

Innovative Computable Quality Control (ICQC) has developed a ground-breaking new machine for performing, well, quality control. Thanks to its novel Deep Intelligence technology, an ICQC quality control (QC) machine can automatically, with 100% accuracy, detect manufacturing errors in any machine in existence, whether it is a coffee machine, an intergalactic space ship, or a quantum computer.

ICQC is now setting up its factory for producing these QC machines. Like any other manufacturing process, some fraction of the produced machines will suffer from malfunctions and these need to be found and discarded. Fortunately, ICQC has just the product for detecting malfunctioning machines!

Obviously, ICQC should not simply use a QC machine on itself, since a malfunctioning machine might incorrectly classify itself as working correctly. Instead, ICQC will take each batch of n machines produced during a day and have them test each other overnight. In particular, during every hour of the night, each of the n QC machines can run a check on one of the other QC machines, and simultaneously be checked by one other QC machine.

If the machine running the check is correct, it will correctly report whether the tested machine is correct or malfunctioning, but if the machine running the check is malfunctioning, it may report either result. If a machine A is used to test a machine B multiple times it will return the same result every time, even if machine A is malfunctioning. The exact testing schedule does not have to be fixed in advance, so the choice of which machines should check which other machines during the second hour of the night may be based on the result of the tests from the first hour, and so on.

ICQC are 100% confident that strictly more than a half of the n QC machines in each batch are working correctly, but the night is only 12 hours long, so there is only time to do a small number of test rounds. Can you help ICQC determine which QC machines are malfunctioning?

For example, consider Sample Interaction 1 below. After the fourth hour, every machine has tested every other machine. For machine 1, only one other machine claimed that it was malfunctioning, and if it was truly malfunctioning then at least 3 of the other machines would claim this. For machine 4, only one other machine claims that it is working, which implies that machine 2 must be malfunctioning since more than half of the machines are supposed to be working. Note that even though machine 4 is malfunctioning, it still happened to produce the correct responses in these specific test rounds.

Interaction

The first line of input contains a single integer b ($1 \leq b \leq 500$), the number of batches to follow. Each batch is independent. You should process each batch interactively, which means the input you receive will depend on the previous output of your program.

The first line of input for each batch contains a single integer n ($1 \leq n \leq 100$), the number of QC machines in the batch. The interaction then proceeds in rounds. In each round, your program can schedule tests for the next hour, by writing a line of the form "`test` $x_1\ x_2\ \ldots\ x_n$" indicating that each machine i should run a test on machine x_i. If $x_i = 0$, then machine i is idle in that round and performs no test. All positive numbers in the sequence must be distinct.

After writing this line, there will be a result to read from the input. The result is one line containing

a string of length n, having a '1' in position i if machine i says that machine x_i is working correctly, '0' if machine i says that machine x_i is malfunctioning, and '–' (dash) if machine i was idle in the round.

When your program has determined which machines are malfunctioning, but no later than after 12 rounds of tests, it must write a line of the form "`answer` S" where S is a binary string of length n, having a '1' in position i if machine i is working correctly, and a '0' if it is malfunctioning.

After writing the answer line, your program should start processing the next batch by reading its number n. When all b batches have been processed, the interaction ends and your program should exit.

Notes on interactive judging:

- *The evaluation is non-adversarial, meaning that the result of each machine testing each other machine is chosen in advance rather than in response to your queries.*
- *Do not forget to flush output buffers after writing. See the Addendum to Judging Notes for details.*
- *You are provided with a command-line tool for local testing, together with input files corresponding to the sample interactions. The tool has comments at the top to explain its use.*

Sample Interaction 1

Read	Write
1 5	
	test 5 4 2 1 3
11011	
	test 4 5 1 3 2
00110	
	test 2 3 4 5 1
01011	
	test 3 1 5 2 4
10100	
	answer 10101

Sample Interaction 2

Read	Write
2 4	
	test 2 3 4 1
1111	
	answer 1111
7	
	test 2 3 4 5 6 7 1
0001100	
	test 0 0 0 0 2 4 0
----11-	
	answer 0101110