Problem B. Ternary

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

This is an interactive problem.

For an n-digit ternary number, Little A designed a very simple encryption algorithm: for each digit $i = 0, 1, \cdots, n-1$, construct a bijection $f_i : \{0,1,2\} \to \{0,1,2\}$. For an n-digit ternary number A, suppose its digits from the most significant to the least significant are:

$$a_{n-1}, a_{n-2}, \cdots, a_0$$

Then its encrypted form from the most significant to the least significant is:

$$f_{n-1}(a_{n-1}), f_{n-2}(a_{n-2}), \cdots, f_0(a_0)$$

To conveniently represent the encryption of each digit, we can denote the bijection $f_i = \{0 \to x, 1 \to y, 2 \to z\}$ (where $x, y, z \in \{0,1,2\}$ and x, y, z are distinct) simply as $f_i = xyz$. For example, $f_0 = \{0 \to 1, 1 \to 2, 2 \to 0\}$ can be represented as $f_0 = 120$.

For instance, if Little A constructs the bijections $f_0 = 021, f_1 = 120, f_2 = 210$ for a 3-digit ternary number, then the encryption of 120 results in 100, and the encryption of 201 results in 012.

Of course, Little A does not want others to know the bijections he constructed, but the encryption algorithm must be used, so Little A will tell you the size of n and allow you to inquire about the results of addition in the encrypted state twice. Your task is to use these 2 inquiries to decipher the n bijections $f_0, f_1, \cdots, f_{n-1}$ constructed by Little A.

Input

The first line contains a positive integer T ($1 \le T \le 10^4$), indicating the number of test cases.

Each test case consists of a single line with a positive integer n ($1 \le n \le 10^5$), representing the number of ternary digits.

It is guaranteed that the sum of n in a single test case does not exceed 10^6.

Interaction Protocol

You can make at most 2 inquiries, after which you must answer Little A's encryption mappings.

For each inquiry, you should provide the two parameters a, b for addition in the format `? a b`, where a, b are both n-digit ternary numbers and must be provided in string format. After flushing the output stream, you need to read a line containing an $n+1$-digit ternary number c in string format, where c is the result of adding the decrypted forms of a and b, followed by encryption, with the most significant digit serving as an overflow flag that is not encrypted. Formally:

$$c = f(f^{-1}(a) + f^{-1}(b))$$

where f is the encryption bijection, and f^{-1} is the inverse mapping of f.

Note that all ternary strings are input from the most significant to the least significant digit!

Note that the highest bit (the n-th bit) of c is treated as an overflow flag and is not encrypted, or you may consider $f_n = 012$.

To answer Little A's encryption mapping, you should provide the n encryption mappings in the format `!` $f_0\ f_1 \cdots f_{n-1}$, where each $f_0, f_1, \cdots, f_{n-1}$ should be one of the strings `012`, `021`, `102`, `120`, `201`, `210`. After

flushing the output stream, the interactor will immediately determine whether your answer is correct, and then proceed to the next test case or end the program without any extra output.

Note that Little A's encryption mappings are fully determined before the inquiries and will not change with the inquiries. In other words, the interactor is not adaptive.

To flush the output stream, you can:

- Use `fflush(stdout)` (if using `printf`) or `cout.flush()` (if using `cout`) in `C/C++`.
- Use `System.out.flush()` in `Java`.
- Use `sys.stdout.flush()` in `Python`.

Example

standard input	standard output
2 3 0210 1102 1 01	? 011 102 ? 010 202 ! 021 120 102 ? 0 1 ! 012

Note

For the first inquiry of the first test case, 011 and 102 decrypt to 102 and 021, respectively, and their sum is 0200, which encrypts to 0210.

For the second inquiry of the first test case, 010 and 202 decrypt to 100 and 221, respectively, and their sum is 1021, which encrypts to 1102.

For the first inquiry of the second test case, 0 and 1 decrypt to 0 and 1, respectively, and their sum is 01, which encrypts to 01.