Collatz Hypothesis and Random Increases

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

This is an interactive problem.

We define the Collatz function collatz(x) which operates on integers as follows: if x is even, then collatz(x) $= \frac{x}{2}$, and otherwise collatz(x) $= 3x + 1$. The famous Collatz hypothesis states that if you start with any positive integer x_0 and construct a sequence $x_1 =$ collatz(x_0), ..., $x_{i+1} =$ collatz(x_i), then the sequence will eventually reach the number one.

The incredible complexity that has kept the hypothesis neither proven nor disproven lies in the very chaotic behavior of the sequence $\{x_i\}$ until it reaches one. Even for very small numbers, one can reach the number one quite late: for example, starting with $x_0 = 9$, we get

$$9 \to 28 \to 14 \to 7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1,$$

and if we start with $x_0 = 27$, we only reach one after 111 applications of collatz(x)!

You are sitting in front of a machine that has a screen and two buttons: red and blue. The screen displays a positive integer (it is guaranteed to be a random integer from 2 to 10^7, inclusive), and you need to turn it into one. The red button is called `collatz`, and it replaces the number x on the screen with collatz(x). The blue button is called `random`, and it replaces x with a random integer from $3x + 1$ to $6x$, inclusive. Pressing the buttons is not free: after each button press, when the screen displays the number x_{i+1}, you need to insert as many tokens into the machine as there are digits in the decimal representation of the number x_{i+1}. For example, in the above process starting from nine, you need to pay for all the digits of the numbers $28, 14, 7, \ldots, 2, 1$; this will cost you 32 tokens.

If you only press the red button, you will reach one, spending an average of 707 tokens. Although the blue button always increases the number on the screen (and significantly), you can speed up the process of reaching one by pressing the blue button at the right moments! Your task is to write a program that spends on average no more than 600 tokens for a single number. To reduce the randomness in the checks, we will provide the program with $t \le 50$ random starting numbers in each test, and you need to turn all of them into ones for a total of $600 \cdot 50 = 30\,000$ tokens.

Interaction Protocol

First, read a line containing an integer t: the number of starting numbers ($1 \le t \le 50$). The jury has t starting numbers x_0, selected uniformly and independently at random from the range $[2; 10^7]$, and you need to turn them all into ones in sequence.

At the beginning of the i-th round, read a line containing a single integer x_0: the next starting number ($2 \le x_0 \le 10^7$). Then you need to transform the number. After each transformation, if the screen shows the number x_j, output a line with either the string "`collatz`" (the letters can be in any case) if you want to replace the number with $x_{j+1} =$ collatz(x_j), or the string "`random`" (the letters can be in any case) if you want to replace the number with x_{j+1} which is a random integer uniformly selected from the range $[3x_j + 1; 6x_j]$. After printing each line, do not forget to flush the output buffer, otherwise, you will likely receive an error of `Idleness Limit Exceeded`:

- `std::cout.flush()` in C++;
- `fflush(stdout)` in C;
- `stdout.flush()` in Python and D;
- `flush(output)` in Pascal and Delphi;
- `System.out.flush()` in Java;
- for help with other languages, refer to the documentation.

After that, read a line containing a single integer x. The following options are possible:

- if $x = 0$, you have run out of tokens;
- if $x = 1$, then $x_{j+1} = 1$: you have successfully completed the round, move on to the next round (for which you should start by reading the next starting number x_0);
- if $x > 1$, then $x_{j+1} = x$: the number on the screen has changed after pressing the button, continue outputting the strings "`collatz`" and "`random`". Note that, during the interaction, x may exceed 2^{64} as well as 2^{128}.

The jury will also send your program the number 0 if you violate the output format and print any string other than "`collatz`" and "`random`". Upon reading 0, your program should immediately terminate to receive a `Wrong Answer` verdict. Otherwise, the verdict can be anything other than `Accepted`.

After reading the t-th number one, terminate the program to receive the `Accepted` verdict.

Example

standard input	*standard output*
2 4 2 1 3 16 8 4 2 1	 Collatz cOLLaTZ RANDOM collatz COLlatz cOLLATz CoLlAtZ

Note

The example is the only test where the starting numbers x_0 are chosen not randomly, but manually.