Problem H. Eggs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Gosha has an egg tray for 16 eggs on the refridgerator door: two rows of sockets, eight sockets in each row, with an immovable separator in the middle. Initially, there are no eggs in the tray.

When there is at least one egg in the tray, Gosha can take one of them and eat it. When there are at most five eggs in the tray, Gosha can go to a grocery store, buy ten eggs there and place them in the tray. Please note that, after he buys new eggs and placing them in the tray, there is always at least one free socket left.

The eggs will spoil if they are not eaten for too long. So, every time Gosha takes an egg to eat, he should pick one of the oldest eggs in the tray: the ones which were bought earliest. He wants to place the newly bought eggs in such a way that, when he sees the positions of all eggs in the tray, he immediately knows which of them are the oldest. Help Gosha devise a strategy: where to place the eggs he buys and which egg to pick for eating so that the picked egg is one of the oldest.

Input

There are two tests in this problem. In the first test, the first line contains the word "`sample`", and in the second test the word "`test`". The answer for the first test is checked for validity only, and on the second test, the strategy is also checked.

Output

Print your strategy as a list of instructions.

Each instruction consists of an initial state (to the left), an action (at the center) and a final state (to the right). Please follow the format shown in the example! The formal output rules are given below the example.

The "`buy`" action must add exactly ten eggs, and the "`eat`" action must remove exactly one egg. A single initial state can occur in the instructions at most twice: at most once with the "`buy`" action and at most once with the "`eat`" action.

In this problem, formatting and the above rules are checked on both tests. However, the strategy checks described below are performed only on the second test.

Gosha uses the strategy as follows. First, he chooses the next possible action: he can eat an egg if there is at least one in the refridgerator, and he can place ten eggs he bought if there are at most five eggs in the refridgerator. After that, Gosha searches for an instruction with the action he chose where the initial state is the actual state of eggs in his refridgerator. Finally, Gosha changes the state to the final state of this instruction.

If there exists a sequence of possible actions such that, eventually, Gosha can not find the instruction he needs, or he picks an egg to eat which is not one of the oldest, the strategy is considered incorrect. If, for every sequence of possible actions, Gosha can find the next instruction, and the egg he eats is always one of the oldest, the strategy is considered correct. Please note that the strategy should account for every sequence of possible actions: when Gosha has two options (eat or buy), both have to be covered.

You are allowed to print instructions with initial states that can not be reached by any sequence of possible actions.

Example

standard input	standard output
sample	****|**** eat ****|**** ...*|**** ....|**** --- ....|.... buy ...*|**** ....|.... ...*|**** ---

Note

Here are the rules for printing instructions. Each instruction takes three lines: the first two describe the states and the action, and the third one consists of three "-" characters (minus, ASCII code 45). In each state, empty sockets are denoted by "." (dot, ASCII code 46), eggs by "*" (asterisk, ASCII code 42), and separators by "|" (vertical line, ASCII code 124). The actions are denoted by words "buy" (add ten new eggs) and "eat" (remove one egg). On the first line, the action is separated from the states by single spaces, and on the second line, the states are separated by five spaces.