2026/2027 SOUTHERN CALIFORNIA REGIONAL
INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 7
Memory Game

This is an interactive problem.

You are going to play the "Memory Game". The rules are simple. There are n matching pairs of cards. The face sides of both cards in a pair show the same image, while each pair's image is unique. The cards are placed face down, shuffled, and laid flat so they can be picked up individually. The cards are numbered from 1 to $2n$ inclusive. In each turn you can pick up any two cards (one at a time, so knowing what you see on the first card can affect which card you will pick up as the second one). If they match, you take them off the board. If not, you return them to their original places, and put them face down again. The card numbers do not change during the game. Your team is to write a program that will match all the cards, using at most $2n - 1$ moves.

Your program is to begin by reading a line with n, the number of card pairs ($2 \leq n \leq 100$). Your program is to then play the game by making moves. Each move consists of picking up two cards. To pick up the first card, output a line with the number of the card your program is selecting. After you print the first number, your program will receive on standard input the name of the image on the selected card, as a string of 1 to 10 upper-case letters. Each image has a unique name. Output another line with the number of the second card your program is selecting. Read the name of the second image on standard input. Repeat making moves until you believe you have won the game by removing all the pairs. At that point, you should output the string "`VICTORY`" on a single line and end your program.

Your program is to track removed pairs and output only valid moves. If you output an invalid move (picking up the same card twice in a single move, picking up a card that has already been removed, or picking up a card with an invalid number), you will immediately lose. If you use more than $2n - 1$ moves, declare victory too early, or forget to declare it at all, you will lose as well.

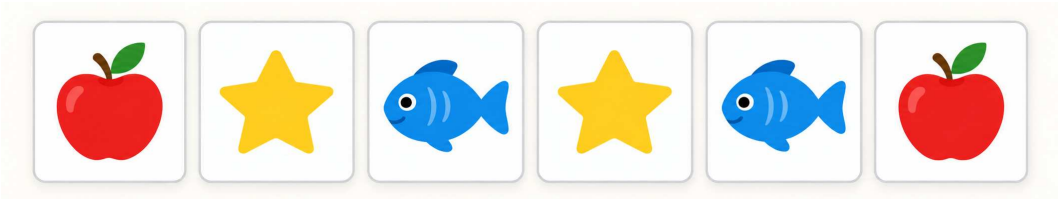


Figure 1. Memory Game cards corresponding to the Sample Interaction.

Problem 7
Memory Game (continued)

Sample Interaction

Read	*Write*
3	
	2
STAR	
	3
FISH	
	1
APPLE	
	6
APPLE	
	4
STAR	
	5
FISH	
	2
STAR	
	4
STAR	
	3
FISH	
	5
FISH	
	VICTORY

Problem 7
Memory Game (still continued)

Judging Notes on Interactive Problems

- In most programming environments, program output is buffered to speed up I/O operations. With interactive problems, it is crucial to make sure the output is actually sent from your program and not simply stored in internal buffers. This typically means flushing the output buffers after each line of output (that is, after each newline character) as follows:
  - In C (or C++ using *cstdio*), you can use `fflush(stdout)`.
  - A C++ output stream is flushed automatically each time you write the `endl` manipulator. When using other means or if you want to be sure, call `cout.flush()`.
  - In Java and Kotlin, the `System.out` stream has so-called "auto-flush" functionality and its buffer is therefore flushed automatically with each newline character. When using other streams or if you want to be sure, invoke the `flush()` method of the stream.
  - In Python, you can use `sys.stdout.flush()`.
- Interactive problems are judged in a way similar to other problems, but there are some differences:
  - When your program attempts to read data, it will wait until more data are available or until the judge program terminates the input (unless you read in a non-blocking way, which is beyond the scope of these notes). Thus, if your program attempts to read more input than can currently be provided (e. g., because you forgot to flush your previous output, or because of some other reason), then the program will stall indefinitely and your submission will get Time Limit Exceeded.
  - As usual, the judgment given to an incorrect submission is the first error discovered, but this does not always mean exactly the same thing as for traditional problems. For instance, if your submission to a traditional problem is too slow on a particular test case, you would get Time Limit Exceeded on that test case. With an interactive problem, the judgment may be Wrong Answer if the solution exhibits an incorrect answer before it runs out of time.
  - The time limit for an interactive problem is how much time your submission may spend; the time spent by the judge program is not counted towards this.
  - Because of timing between the interactive judge program and your submission, judgments for incorrect submissions to interactive problems are not necessarily deterministic. We can guarantee the following: a judgment of Wrong Answer means that your submission produced incorrect output, and a judgment of Run Time Error means that your submission returned a non-zero exit code. If your submission does both, you may receive either judgment.