

## 6. Covert Channel (kanal)

1 sec / 3 sec

100 points

Agent Wild Duck is on a mission to a top-secret North Korean factory. The factory has very strong security measures and all communication with the outside world is restricted. For simplicity, the information transmitted by the agent from the factory (daily) is a single integer  $S$  ( $0 \leq S < M \leq 10^9$ , where  $M$  is a fixed parameter in each test session).

Every day, a train of exactly  $N$  wagons ( $1 < N \leq 1000$ ) leaves the factory. All wagons have numbers (integers from 0 to  $10^9$ , written on the wagons). Agent Wild Duck can influence the work of the train-assembler — he can determine the order in which the wagons are coupled. Reconnaissance satellites can identify the numbers on the wagons of the train leaving the factory.

Implement the encoding and decoding of information for the described covert channel. Efficiency is also important — we want the information to be identifiable from a minimal number of wagons at the beginning of the train.

The wagon numbers do not have to be unique and two wagons with the same number are indistinguishable. However, it can be assumed that there are always enough different numbers in use for the task to be solvable for the given values of  $N$  and  $M$  — for example, it is certainly not possible for all wagons to have the same number.

**Interaction.** This is an interactive task where the program must encode and decode multiple messages in one session. Two instances of the program will be run in different modes.

Data encoding:

- The first line of input contains the text ‘ENC’.
- The second line of input contains the integers  $M$  and  $N$ .
- The third line of input contains  $N$  integers: the numbers of the wagons.
- Repeat: Read the number  $S$  to be encoded from the next line of input. If  $S \geq 0$ , output the permutation of the wagons ( $N$  integers) to the next line of output. If  $S = -1$ , the program must terminate.

Changing the wagon numbers, using some wagons multiple times, or not using all of them is forbidden.

Data decoding:

- The first line of input contains the text ‘DEC’.
- The second line of input contains the integers  $M$  and  $N$ .
- To get the next wagon number, output ‘R’ as the next line of output and read an integer from the next line of input.
- To give an answer, write an integer as the next line of output. The system will respond to the next ‘R’ command with the number of the first wagon of the next train. The wagon number  $-1$  means there are no more tests, the program must terminate.

Issuing ‘R’ more than  $N$  times in a row is an error and the program’s testing will be terminated. The decoded numbers (trains) might not be in the same order as the encoder processed them.

**Grading.** In this task, each session is evaluated separately.

- A session has up to 1000 tests.
- Each session is evaluated separately, in comparison with the solutions of other contestants. The number of points earned is inversely proportional to the number of ‘R’ commands used by the decoder. The best contestant gets the maximum score.

- A program that violates the communication protocol or exceeds the time limit gets 0 points. The time limit applies to the total running time of the encoder and decoder.

The following additional conditions apply to the distribution of points:

0. (0 points) The example in the task description.
1. (20 points)  $M = 2$ .
2. (20 points)  $M < N$ .
3. (10 points) The wagons are always numbered  $1, \dots, N$ .
4. (10 points) The wagon numbers are unique.
5. (40 points) No additional constraints.

Encoder	Input	Output
	ENC	
	2 5	
	1 2 5 4 3	
	0	
		1 2 3 4 5
	1	
		2 1 5 4 3
	-1	
Decoder	Input	Output
	DEC	
	2 5	
		R
	2	
		R
	1	
		1
		R
	1	
		R
	2	
		0
		R
	-1	

**Remark.** To ensure that the data output by the program reaches the testing system, the output buffer must be flushed after each line:

Programming language	Command
C++	<code>cout &lt;&lt; ... &lt;&lt; endl;</code> or <code>cout &lt;&lt; ... &lt;&lt; "\n" &lt;&lt; flush;</code>
Python	<code>print(..., flush=True)</code> or <code>sys.stdout.write(...)</code> <code>sys.stdout.flush()</code>