# Problem L. Business Semiconductor Units

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

*Business Semiconductor Units* (BSU) is a large international corporation that focuses on selling fast and reliable computers to business clients. Recently, they have decided to develop a new processor model which will work even faster and more reliably than its predecessors.

The R&D department of the company is responsible for designing the instruction set and processor architecture. After the deadline, they should demonstrate the working prototype to the head of the company. Unfortunately, the whole department was playing *Minecraft* most of the time instead of doing their job, so the presented prototype supports only three simple instructions.

Let's take a closer look at their masterpiece. The new processor has 16 registers named from `r0` to `r15`, each of them can store an unsigned 16-bit integer. There is also main memory consisting of $2^{16}+1$ eight-bit cells.

The *program* for this processor is a sequence of instructions. The instructions are executed sequentially, neither jumps nor loops are supported. The processor executes the same sequence of instructions 5000 times. That is, the following procedure is repeated 5000 times: go over the instructions from the start to the end and execute them.

Below you can see the list of available instructions. For clarity, let's call $x \bmod 2^8$ the *lower* part of the number $x$, and $\left\lfloor \frac{x}{2^8} \right\rfloor$ the *upper* part of the number $x$. The number in the $i$-th main memory cell is denoted $mem_i$.

- `imm r, b`: load the constant number $b$ ($0 \le b < 2^{16}$) into the register named $r$;
- `ld x, y`: suppose that the register named $y$ stores the number $b$. Then, the number $mem_{b+1} \cdot 2^8 + mem_b$ is put into the register $x$;
- `st x, y`: suppose that the register named $x$ stores the number $a$, and the register $y$ stores the number $b$. Then, the lower part of $b$ is put into $mem_a$, and the upper part of $b$ is put into $mem_{a+1}$.

As you can see, the instruction set is pretty lean, and the R&D department is unsure whether the processor is capable of doing anything non-trivial or not. To make it run some useful programs, they hired you and gave you an assignment. Now, you need to write a program for the new processor that multiplies $n$ non-negative 16-bit numbers modulo $2^{16}$.

## Input

This problem has no input data.

## Output

Output the required program in the following format.

The first line must contain an integer $s$, the number of instructions in your program ($1 \le s \le 10^5$).

Each of the following $s$ lines must contain a processor instruction. The format of instructions is described above. Be careful and follow the format strictly. All the register names must be valid (that means, from `r0` to `r15`).

## Interaction Protocol

Technically, this problem is an output-only interactive problem. [Sounds weird, isn't it? :)]

In each test, the interactor first reads the instructions you wrote to the output. Next, it reads the integer $n$ and $n$ integers $a_i$ from the test ($1 \le n \le 4000$, $0 \le a_i < 2^{16}$). The number $n$ is placed into the register `r0`. For each $1 \le i \le n$, the lower part of $a_i$ is placed into $mem_{2 \cdot i-2}$, and the upper part is placed into $mem_{2 \cdot i-1}$. All other registers and memory cells are zeroed initially.

Then, the interactor executes your program. The instructions are performed sequentially, and the execution of the program is performed exactly 5000 times.

After that, the interactor reads your answer from the register `r0` and compares it with $a_1 \cdot a_2 \cdot \ldots \cdot a_n$ modulo $2^{16}$.

## Example

| standard input | standard output |
|---|---|
| | 4<br>imm r3, 42<br>imm r1, 6<br>st r3, r1<br>ld r0, r3 |

## Note

The output in the example does not multiply integers, so submitting this program will give you the "`Wrong Answer`" verdict. The program is only provided to illustrate the output format.