Problem K. Kitten's Computer

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Kitten recently planned to build a computer of his own. The computer has 400 registers, each of which can store a 64-bit binary integer, that is, an integer in the range $[0, 2^{64} - 1]$. The value stored in the i-th ($i \in [1, 400]$) register is denoted as a_i. This computer supports 7 assembly instructions:

- `SET i j` : Let $a_i := a_j$.
- `XOR i j k` : Let $a_i := a_j \oplus a_k$ ($\oplus$ is the bitwise XOR operation).
- `AND i j k` : Let $a_i := a_j \,\&\, a_k$ ($\&$ is the bitwise AND operation).
- `OR i j k` : Let $a_i := a_j \,|\, a_k$ ($|$ is the bitwise OR operation).
- `NOT i j` : Let $a_i := \sim a_j$ ($\sim$ is the unary bitwise NOT operation).
- `LSH i x` : Shift a_i left by x bits. The vacant bit-positions are filled with 0.
- `RSH i x` : Shift a_i right by x bits. The vacant bit-positions are filled with 0.

Note that you have to ensure that $1 \le i, j, k \le 400$ and $0 \le x < 64$.

You may think that this computer is not powerful enough, but the kitten's computer is not an ordinary computer! This computer has a powerful parallel computing method that can compute all non-interfering instructions simultaneously.

Formally, let us track $t_1, t_2, \ldots, t_{400}$, denoting the times when the register values were assigned. Initially, all t_i are zeroes. Whenever you execute a command, if it requires $a_{j_1}, a_{j_2}, \ldots, a_{j_n}$ as arguments to calculate, and outputs the result to a_i, then assign t_i to $\max\{t_{j_1}, t_{j_2}, \ldots, t_{j_n}\} + 1$. The *runtime* of your program is the maximum value of all t_i generated during the sequential execution of all instructions.

Today, Kitten wants to use his computer to design a calculator. This calculator is used to quickly calculate the multiplication of 64-bit unsigned integers. At the beginning, registers a_1 and a_2 are set to two 64-bit unsigned integers x and y, respectively, while the other registers are set to 0. You need to help Kitten design a series of instructions for his program so that the final value of a_1 is the result of multiplying x and y, modulo 2^{64}.

Kitten requires that the total number of your instructions does not exceed 100 000, and the *runtime* of your program does not exceed 70.

Input

There is no input for this problem.

Output

Output any number of lines (from 0 to 100 000), each containing exactly one instruction formatted as shown above.

Example

standard input	standard output
<no input>	NOT 2 1 RSH 2 63 NOT 3 1 RSH 3 62 NOT 4 1 RSH 4 61 LSH 2 1 LSH 3 9 LSH 4 3 OR 5 2 3 OR 1 5 4

Note

The example output does not solve the problem, it is given only to demonstrate the format.

When checking your output, the checker will perform the following checks.

1. If your output exceeds 100 000 lines, return WA and exit immediately.
2. If your output contains an illegal instruction, return WA and exit immediately.
3. Perform the following process 5000 times:
   (a) Given are two 64-bit unsigned integers x and y.
   (b) Clear all registers to zero and make $a_1 = x$ and $a_2 = y$.
   (c) Execute your program.
   (d) If the *runtime* exceeds 70, return WA and exit immediately.
   (e) Check if the value of a_1 is $(x \cdot y) \bmod 2^{64}$. If not, return WA and exit immediately.
4. Return OK and exit immediately.

Note that the checker will only check the register a_1. The final values of all other registers can be arbitrary.

The 5000 pairs of x and y for the checker are fixed in advance.