Problem H. Fast Debugger

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

Recently you received an old computer. This old computer has an 8-bit CPU with four registers: `ax`, `bx`, `cx`, `dx`, and only supports some simple instructions. For those who are not familiar with assembly language, here is a programming guide.

This CPU has four 8-bit registers: `ax`, `bx`, `cx`, `dx`. You can treat them as variables storing integers within $[0, 255]$.

Only three kinds of bitwise operation are supported by this CPU, bitwise-or/and/xor. Each bitwise operation has two kinds of instructions.

Type 1: Both operands are registers, written as "`or r1 r2`", where both `r1` and `r2` are one of the register names `ax`, `bx`, `cx`, `dx` (`r1` and `r2` may refer to the same register). This instruction will set the value of `r1` to the result of `r1 bitwise-or r2`. (Similarly, bitwise-and and bitwise-xor instructions are written as "`and r1 r2`" and "`xor r1 r2`").

Type 2: One of the operands is immediate, written as "`ori r imm`", where `r` is one of the register names `ax`, `bx`, `cx`, `dx`, and `imm` is a constant in $[0, 255]$ given in the instruction. This instruction will set the value of `r` to the result of `r bitwise-or imm`. (Similarly, bitwise-and and bitwise-xor instructions are written as "`andi r imm`" and "`xori r imm`").

Loops are not supported by the CPU, but the assembler implemented an easy loop for programmers. If the assembler sees a "`repeat` m" statement, it will automatically repeat the contents of the repeat block, a total of m times. The format is shown below.

```
repeat m
<repeat block>
end
```

Here, m is a constant in $[2, 255]$ given in the statement, and `<repeat block>` consists of one or more statements that can be either bitwise instructions or repeat–end statements.

Now you want to write a simulator on your new laptop which is much faster than the old computer.

Your simulator will be given a valid program and q queries. Each query consists of five integers k, a_0, b_0, c_0, d_0. Initially, the registers are set to the given values: `ax` $= a_0$, `bx` $= b_0$, `cx` $= c_0$, `dx` $= d_0$. You should output the values of registers `ax`, `bx`, `cx`, `dx` after the program executes k bitwise instructions.

Input

The first line of input contains two integers n and q ($1 \le n \le 12\,000$; $1 \le q \le 10\,000$), denoting the number of instructions and the number of queries.

Then follow n lines. Each line is an instruction. The format is described above.

Each of the following q lines contains five integers k, a_0, b_0, c_0, d_0 ($1 \le k \le 10^9$; $0 \le a_0, b_0, c_0, d_0 \le 255$), denoting a query for the value of registers after evaluating k bitwise operations. It is guaranteed that the program does not terminate before executing k bitwise instructions.

Output

Output q lines. The i-th line must contain four integers a_k, b_k, c_k, d_k, denoting the values of registers `ax`, `bx`, `cx`, `dx` after the program executes k bitwise instructions.

Example

standard input	*standard output*

`6 2`
`repeat 5`
`xor ax bx`
`xori ax 3`
`and cx ax`
`xor cx dx`
`end`
`10 1 2 4 3`
`8 4 1 2 3`
	`0 2 2 3`
	`4 1 3 3`