Problem A. Random Number Generator Manipulation

Input:	`stdin`
Output:	`stdout`
Time Limit:	0.5 second
Memory Limit:	1024 MB

In gacha games, random number generation (or RNG for short) plays a lot of part. After failing a 1% gacha 200 times in a row, Ema became salty and decided that she has had enough, now she want revenge on the game.

Due to her love hate relationship with the game, she decided it would be enough to mess with the RNG system. Of course, all the gacha stuff are done on the server's side, but being able to control local RNG like getting skills to activate every time would be enough to gain her a substantial advantage over the game.

Being the 31337 H4X0R with massive skillz that she is, Ema was able to determine that the the RNG function looks as follow:

- Take an unsigned m bits binary value as input (from 0 to $2^m - 1$)
- Perform a program on said value
- Return the output value

Ema also managed to recover the instructions that make up the function. According to her, there are n instructions that will be executed in order. The instructions are as follow:

- `add x` ($0 \le x < 2^m$): Add x to the current result. Note that unsigned values wrap around, so if the result is greater than or equal to 2^m, it will be taken modulo 2^m
- `mul x` ($0 \le x < 2^m$): Multiply the result by x, warps around the same way as `add`
- `mod x` ($1 \le x < 2^m$): Modulo the current result by x.
- `if x y` ($0 \le x \le y < 2^m$): If the value is inside the range [x, y] (inclusively), execute the instructions until the matching `end_if`
- `end_if`: Mark the end of an if block.

There can be nested `if end_if` blocks, but all `if` will have a matching `end_if`. For example, the following is a function for generating random numbers when $m = 5$:

```
mul 13
add 17
mod 30
if 0 9
    if 0 4
        add 10
    end_if
    if 5 9
        add 5
    end_if
end_if
```

The above function will generate a random number between 10 and 29, with the values in the range [10, 14] being 3 times more likely than values outside of it (assuming the random number is good enough).

By tampering with the (emulated) CPU, Ema can make some instructions of type `add`, `mul`, or `mod` return the input value instead of the correct result, effectively letting her skip these instructions.

By skipping instruction in an optimal way, Ema can potentially make the function return a value favorable to her. However, there are some instruction that Ema cannot tamper with (or the game will detect that something is wrong and invalidate her game play). Please help her calculate which output is possible from which input so she can further her revenge plan.

Input

- The first line contain 2 positive integers n and m, ($1 \le n \le 10^5, 1 \le m \le 5$)

- The following n lines describe the instructions, they have one of the following form:
  - `add x b`: Add x to the current result (modulo 2^m).
  - `mul x b`: Multiply the current result by x (modulo 2^m).
  - `mod x b`: Modulo the current result by x.
  - `if x y`: Execute until until the corresponding `end_if` if the current value is in the range $[x, y]$.
  - `end_if`: Mark the end of an `if` block.
- Constrains on the input:
  - $0 \le x, y < 2^m$
  - $0 \le b \le 1$. If $b = 1$, Ema cannot tamper with that instruction.
  - For the `mod` instructions, $x > 0$.
  - For `if`, $x \le y$.
  - Each `if` has a matching `end_if`, and no `if - end_if` block will be empty.

Output

- Print 2^m lines, each line is a binary string of length 2^m.
- The j-th character on the i-th line (counting from 0) should be `'1'` if given the input value i, there is a way to manipulate the function to output j, otherwise, it should be `'0'` (output without the quotes).

Examples

stdin	stdout
1 1 add 1 0	11 11
1 1 add 1 1	01 10
5 2 add 1 0 if 0 1 add 1 0 end_if add 1 1	0111 0011 1001 1110

Subtask 1 (5 points)

$n \le 100, m \le 5$

Subtask 2 (7 points)

$n \le 1\,000, m \le 5$

Subtask 3 (11 points)

For all `add`, `mul`, `mod`, $b = 1$ (Ema cannot tamper with anything).

Subtask 4 (13 points)

For all `add`, `mul`, `mod`, $b = 0$ (Ema can tamper with everything).

Subtask 5 (17 points)

There is no `if` (and `end_if`) instructions.

Subtask 6 (23 points)

$m \le 3$

Subtask 7 (24 points)

No additional constraints