Problem J. LaLa and Magical Beast Summoning

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

LaLa is about to cast a magical beast summoning magic.

The first thing LaLa do is creating a **summoning field**, which has 3 constants associated with it: **nullity** M, **elasticity** E, and **viscosity** V. Such summoning field is denoted by $\mathcal{F}(M, E, V)$

A magical beast summoning magic is performed over a **summoning cell** within the summoning field, which is square-shaped and is associated with 3 variables: **side length** L, **agility** A, and **intelligence** I. Such summoning cell is denoted by $\mathcal{C}(L, A, I)$.

$\mathcal{C}(L, A, I)$ is in a **null state** if $L = 0$. Otherwise, it is in a **positive state**.

The **density** of $\mathcal{C}(L, A, I)$ in positive state is defined as $(A \times I)/L^2$.

The problem of determining whether a magical beast summoning magic will succeed requires very heavy computation involving solving a system of 9999999999-th order partial differential equations over 9999999999999999 variables. Fortunately, LaLa already did all the math for you!

The magical beast summoning magic over $\mathcal{C}(L, A, I)$ within $\mathcal{F}(M, E, V)$ succeeds if and only if the function valid(M, E, V, L, A, I) defined by the pseudocode in the note section returns true. We'll call such summoning cell **valid**.

Sometimes, LaLa isn't satisfied with the set of summoning cells she has, and wants to generate new ones by combining them. The problem of determining the result of combination of two valid summoning cells $C_0 = \mathcal{C}(L_0, A_0, I_0)$ and $C_1 = \mathcal{C}(L_1, A_1, I_1)$ within $F = \mathcal{F}(M, E, V)$ requires another heavy computation, but thankfully, LaLa already did all the math for you again!

The result of combining two such cells C_0 and C_1 within F, denoted by $\text{Combine}_F(C_0, C_1)$, is given by the function combine$(M, E, V, L_0, A_0, I_0, L_1, A_1, I_1)$ defined by the pseudocode in the note section, which returns a triple L_2, A_2, I_2 satisfying $\mathcal{C}(L_2, A_2, I_2) = \text{Combine}_F(C_0, C_1)$. Here, it can be proved that $\text{Combine}_F(C_0, C_1)$ is also valid. Note that swapping the order of C_0 and C_1 affects the result.

The result of combining $K \geq 3$ cells $C_0, \cdots, C_{K-1}$ within F is given recursively by

$$\text{Combine}_F(C_0, \cdots, C_{K-1}) = \text{Combine}_F(\text{Combine}_F(C_0, \cdots, C_{K-2}), C_{K-1})$$

For the sake of completeness, we define $\text{Combine}_F(C) = C$.

LaLa **is aware of a very special property about the combining operation that allows her to efficiently solve the range density query problem below.** Can you figure it out?

You're given a summoning field $F = \mathcal{F}(M, E, V)$ and an array of N valid summoning cells

$$C_0 = \mathcal{C}(L_0, A_0, I_0), \cdots, C_{N-1} = \mathcal{C}(L_{N-1}, A_{N-1}, I_{N-1})$$

within F. Write a program that processes the following two types of Q queries:

- `1 i L A I`

  - Set $C_i \leftarrow \mathcal{C}(L, A, I)$.

- `2 l r`

  - Let $R = \text{Combine}_F(C_l, \cdots, C_{r-1})$. If R is in the null state, print a single integer -1. Otherwise, print the density of R, modulo M. Here, an irreducible fraction p/q, where p is a non-negative integer and q is a positive integer not divisible by M, modulo M is defined to be the unique integer $p \times q^{-1} \bmod M$ where q^{-1} is the multiplicative inverse of q modulo M. It can be proved that if R is in the positive state, the denominator of the density of R as an irreducible fraction is not divisible by M within the constraints of this problem.

Input

The input is given in the following format:

$$\begin{array}{lll} M & E & V \\ N & & \\ L_0 & A_0 & I_0 \\ & \vdots & \\ L_{N-1} & A_{N-1} & I_{N-1} \\ Q & & \\ q_0 & & \\ \vdots & & \\ q_{Q-1} & & \end{array}$$

Here, q_i denotes the i-th query, and is given in the format described in the statement.

The input satisfies the following constraints:

- All the numbers in the input are integers.
- M is a prime such that $900\,000\,000 \le M \le 1\,000\,000\,000$
- $1 \le E, V \le 100$
- $1 \le N, Q \le 100\,000$
- $0 \le L_i, A_i, I_i < M$ for all integers $0 \le i < N$
- $\mathcal{C}(L_i, A_i, I_i)$ within $\mathcal{F}(M, E, V)$ is valid for all integers $0 \le i < N$.
- For each query `1 i L A I`, $0 \le i < N$, $0 \le L, A, I < M$, and $\mathcal{C}(L, A, I)$ is valid within $\mathcal{F}(M, E, V)$.
- For each query `2 l r`, $0 \le l < r \le N$

Output

For each query of the second type, print its answer in a single line.

Example

standard input	standard output
998244353 1 2 3 2 998244352 3 4 998244351 6 4 929561374 68682991 7 2 0 2 2 0 3 2 1 3 1 1 6 9 998244350 2 0 2 2 0 3 2 1 3	-1 748683259 156877648 748683265 499122176 342244251

Note

The following pseudocode defines the validity of summoning cells and the Combine operation.

Both functions do not modify their arguments

```
function VALID(M, E, V, L, A, I)
    if min(L, A, I) < 0  or  M ≤ max(L, A, I) then
        return False
    end if
    /*Every operations and comparisons below are done mod M*/
    if L = 0  and  (A + I ≠ 0  or  A = I) then
        return False
    end if
    return A^3 − A^2 L + 3A^2 I + EAL^2 + L^3 V + 2ALI + EL^2 I + 3AI^2 − LI^2 + I^3 ≠ 0
end function
```

```
function COMBINE(M, E, V, L0, A0, I0, L1, A1, I1)
    Ensure VALID(M, E, V, L0, A0, I0)
    Ensure VALID(M, E, V, L1, A1, I1)
    /*Every operations and comparisons below are done mod M*/
    if L1 = 0 then
        return L0, A0, I0
    end if
    if L0 = 0 then
        return L1, I1, A1
    end if
    B0 ← (A0 + I0) · L1, B1 ← (I1 + A1) · L0
    C0 ← (A0 − I0) · L1, C1 ← (I1 − A1) · L0
    if B0 = B1 then
        if C0 + C1 = 0 then
            return 0, 3, −3
        end if
        Sum ← A0 + I0, Dif ← A0 − I0
        B ← 3 · Sum^2 + E · L0^2
        C ← 2 · Dif · L0
        D ← 2 · C · Sum · Dif
        E ← B^2 − 2 · D
        X ← C · E, Y ← B · (D − E) − 2 · C^2 · Dif^2
        return 2 · C^3, X + Y, X − Y
    else
        B ← B0 − B1, C ← C0 − C1, D = L0 · L1
        E ← C^2 · D − B^2 · (B0 + B1)
        X ← B · E, Y ← C · (B0 · B^2 − E) − C0 · B^3
        return 2 · B^3 · D, X + Y, X − Y
    end if
end function
```

The author has attached a C++ implementation which will get "Time Limit Exceeded" verdict upon submission, but it will always print the correct answer within finite time. You may reuse some part of the implementation on your submission. (If you're reading the printed version, you may find the attachment at the bottom of the statement on the eolymp site.)