7472 Generators

Little Roman is studying *linear congruential generators* — one of the oldest and best known pseudo-random number generator algorithms. Linear congruential generator (LCG) starts with a non-negative integer number x_0 also known as *seed* and produces an infinite sequence of non-negative integer numbers x_i $(0 \le x_i < c)$ which are given by the following recurrence relation:

$$x_{i+1} = (ax_i + b) \bmod c$$

here a, b, and c are non-negative integer numbers and $0 \le x_0 < c$.

Roman is curious about relations between sequences generated by different LCGs. In particular, he has n different LCGs with parameters $a^{(j)}$, $b^{(j)}$, and $c^{(j)}$ for $1 \le j \le n$, where the j-th LCG is generating a sequence $x_i^{(j)}$. He wants to pick one number from each of the sequences generated by each LCG so that the sum of the numbers is the maximum one, but is not divisible by the given integer number k.

Formally, Roman wants to find integer numbers $t_j \ge 0$ for $1 \le j \le n$ to maximize $s = \sum_{j=1}^{n} x_{t_j}^{(j)}$ subject to constraint that $s \bmod k \neq 0$.

Input

The input file contains several test cases, each of them as described below.

The first line of the input file contains two integer numbers n and k $(1 \le n \le 10000,\ 1 \le k \le 10^9)$.

The following n lines describe LCGs. Each line contains four integer numbers $x_0^{(j)}$, $a^{(j)}$, $b^{(j)}$, and $c^{(j)}$ $(0 \le a^{(j)}, b^{(j)} \le 1000,\ 0 \le x0 < c^{(j)} \le 1000)$.

Output

For each test case, the output must follow the description below.

If Roman's problem has a solution, then write on the first line of the output file a single integer s — the maximum sum not divisible by k, followed on the next line by n integer numbers t_j $(0 \le t_j \le 10^9)$ specifying some solution with this sum.

Otherwise, write to the output file a single line with the number '`-1`'.

Notes:

In the first example, one LCG is generating a sequence 1, 2, 3, 4, 5, 0, 1, 2, ..., while the other LCG a sequence 2, 3, 2, 3, 2, ...

In the second example, one LCG is generating a sequence 0, 2, 0, 2, 0, ..., while the other LCG a sequence 2, 4, 2, 4, 2, ...

Sample Input

```
2 3
1 1 1 6
2 4 0 5
2 2
0 7 2 8
2 5 0 6
```

Sample Output

```
8
4 1
-1
```