Problem C. Buggy Painting Software II

This is a communication problem.

Well... after surviving the nightmarish stutter in **Problem B. Buggy Painting Software I**, Sulfox soon faces a far more serious bug — his painting software saves any color image in black-and-white format.

Here, a color image supports m distinct colors labeled 1 through m, and can be viewed as a sequence of length $3m$ composed of color labels, where each color appears **exactly three times**. A black-and-white image supports only black (labeled 0) and white (labeled 1), and can be viewed as a binary sequence. When saving a color image, the software chooses a non-empty bipartition (S_0, S_1) of $S = \{1, 2, \ldots, m\}$ (so $S_0 \cup S_1 = S$, $S_0 \cap S_1 = \varnothing$, and both S_0 and S_1 are non-empty) and converts the original sequence into a binary sequence of the same length by mapping every color in S_0 to 0 and every color in S_1 to 1. The resulting binary sequence will be the saved black-and-white image.

To outsmart this bug, Sulfox wants to create n color images with respective integer feature values $x_1, x_2, \ldots, x_n$ (values may repeat) ranging from 1 to m. Your task is to help Sulfox design these color images so that each image's feature remains identifiable even after the color loss. More specifically, the i-th color image ($1 \le i \le n$) must satisfy that, no matter how the non-empty bipartition (S_0, S_1) is chosen, its feature value x_i can always be recovered from the saved black-and-white image.

To verify the correctness of your design, your program will be **run twice** on each test: first for constructing n color images, and then for recovering the original feature values from n black-and-white images. The second run will receive exactly the n binary sequences mapped from each of the n sequences you output in the first run by independently chosen non-empty bipartitions. Apart from that, the second run will receive no additional information from the first run. Your solution passes a test if you correctly recover the corresponding feature value from each black-and-white image.

Note that the applied bipartitions are unpredictable and may be **adaptive** to your design, and different bipartitions may apply to the color images with the same feature value.

Input

The first line contains three integers op ($op \in \{1, 2\}$), n, and m ($2 \le n, m \le 5 \times 10^5$, $nm \le 10^6$), denoting which run of the program it is, the number of color images, and the number of distinct colors, respectively. It is guaranteed that n and m remain consistent across both runs.

If $op = 1$, your program must construct the color images. The next n lines follow, where the i-th line contains a single integer x_i ($1 \le x_i \le m$), representing the feature value for the i-th image to be designed.

If $op = 2$, your program must recover the feature values. The next n lines follow, each describing a black-and-white image. Each line contains $3m$ space-separated integers of 0 or 1, forming a binary sequence that results from one of your designed color images. These n sequences are given in an **arbitrary order** and do not necessarily correspond to the order in which the feature values were given in the first run.

Output

If $op = 1$, output n lines, where the i-th line describes the designed color image for the i-th feature value x_i from the input. Each line contains $3m$ space-separated integers. The sequence on each line must be a valid color image, i.e., a sequence where each integer from 1 to m appears exactly three times.

If $op = 2$, output n lines. For each of the n binary sequences given in the input, output a single line containing one integer, representing the original feature value recovered from that image. The order of your output must correspond to the order of the binary sequences in the input.

Examples

standard input	standard output
1 2 3 1 2	1 1 1 2 2 2 3 3 3 1 2 3 1 2 3 1 2 3
2 2 3 1 1 0 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0	2 1

Note

The example shows two runs of a certain solution on the sample case.

In the first run, a simple (and incorrect for general cases) strategy is used: feature value 1 is designed as consecutive blocks of three identical colors, and feature value 2 is designed as a repeating pattern of three distinct colors.

In the second run, the first binary sequence (from bipartition $(S_0 = \{3\}, S_1 = \{1, 2\})$) is identified by its repeating pattern of three to recover feature value 2, while the second binary sequence (from bipartition $(S_0 = \{1, 3\}, S_1 = \{2\})$) is identified by its consecutive blocks of three to recover feature value 1. Since the judge provides these sequences in a swapped order, the output is 2 and then 1.