## Problem 5 – Bits Inverter

Once Teodor played the following game: he started from a sequence of white and black balls and flipped the color of the 1st ball, then the color of the 4th ball, then the color of the 7th ball, etc. until the last ball. Flipping was performed by replacing a black ball with a white ball and vice versa. Teodor was a smart mathematician so he wanted to generalize his game in a formal way. This is what he invented.

You are given a **sequence of bytes**. Consider each byte as sequences of exactly 8 bits. You are given also a number **step**. Write a program to invert the bits at positions: **1**, **1 + step**, **1 + 2\*step**, ... Print the output as a sequence of bytes.

Bits in each byte are counted from the leftmost to the rightmost. Bits are numbered starting from 1.

### Input

* The input data should be read from the console.
* The number **n** stays at the first line.
* The number **step** stays at the second line.
* At each of the next **n** lines **n** bytes are given, each at a separate line.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output should be printed on the console. Print exactly **n** bytes, each at a separate line and in range [0..255], obtained by applying the bit inversions over the input sequence.

### Constraints

* The number **n** will be an **integer** number in the range [1…100].
* The number **step** will be an **integer** number in the range [1…20].
* The **n numbers** will be integers in the range [0…255].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  11  45  87 | 173  71 | We have the following sequence of 16 bits (2 bytes):  **0**0101101 010**1**0111  We invert the bits 1 and 12 (step=11). We get:  **1**0101101 010**0**0111 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  5  45  87  254 | 169  118  246 | We have the following sequence of 24 bits (3 bytes):  **0**0101**1**01 01**0**1011**1** 1111**1**110  We invert the bits 1, 6, 11, 16 and 21 (step=5). We get:  **1**0101**0**01 01**1**1011**0** 1111**0**110 |