## Problem 5 – Neuron Mapping

Perhaps you have heard of EyeWire, an online project by the Massachusetts Institute of Technology (MIT), where volunteers color sequences of cross sections of brain cells, and help the AI analyze their structure. The project is extremely promising, and has already managed to create the first complete map of a human brain. There is one problem, however – even with the thousands of volunteers, it's still too slow. Let's help them!

In this problem, you will implement the first part of a solution that can recognize neurons from 2D images. You will be given a grid of black and white pixels, represented by '1' and '0' bits, and your task will be to produce an image showing which pixels represent the inside parts of a neuron.

### Problem Description

You will read a sequence of **32 bit integers**, representing the image grid. Every '1' bit is a black pixel, representing a **neuron boundary**, and every '0' bit is either space **between the neurons** or **inside a neuron**. Neurons don't touch or intersect, and have relatively simple shapes. Your task is to produce an image in the same format, where the black pixels are inside the neurons.

### Full Example

('0' bits represented as dots for ease of reading)

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Input as Grid** | **Output** | **Output as Grid** |
| 480  272  224  16252928  50593792  33685504  67239936  67174400  33587200  16809984  16973824  8650752  7864320  0 | .......................1111.....  .......................1...1....  ........................111.....  ........11111...................  ......11.....1..................  ......1.......1.................  .....1........1.................  .....1.........1................  ......1.........1...............  .......1........1...............  .......1......11................  ........1....1..................  .........1111...................  ................................ | 0  224  0  0  16252928  33292288  66846720  66977792  33488896  16711680  16515072  7864320  0  0 | ................................  ........................111.....  ................................  ................................  ........11111...................  .......1111111..................  ......11111111..................  ......111111111.................  .......111111111................  ........11111111................  ........111111..................  .........1111...................  ................................  ................................ |

Note that the neuron **boundaries are excluded** from the output image (they are represented by '0' bits, just like the space between the neurons).

The neurons **don't intersect or touch**, and don't intersect the boundaries of the image (all neurons are **whole**). The neurons are located on **distinct rows** (i.e. each row contains pixels of either zero or one neurons). All neuron have **convex** shapes. There is no garbage on the input picture - all '1' bits are part of the neuron boundaries.

### Input

The input data should be read from the console. There will be 0 or more integers coming in from the console, one per line, ending with -1.

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

### Output

The output data should be printed on the console.

You should print the same number of non-negative integers as you read in (except the last -1), each output integer representing a row of the output image.

### Constraints

* All input integers are non-negative and will fit into 4 bytes.
* The number of input integers are less than 32.
* Allowed work time for your program: 0.1 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 3758096384  2684354560  3758096384  0  0  0  0  -1 | 0  1073741824  0  0  0  0  0 | 0  0  0  16252928  50593792  33685504  67239936  67174400  33587200  16809984  16973824  8650752  7864320  0  -1 | 0  0  0  0  16252928  33292288  66846720  66977792  33488896  16711680  16515072  7864320  0  0 | 480  272  224  16252928  50593792  33685504  67239936  67174400  33587200  16809984  16973824  8650752  7864320  0  -1 | 0  224  0  0  16252928  33292288  66846720  66977792  33488896  16711680  16515072  7864320  0  0 |