Vision Program

You are implementing a vision program for a robot. Each time the robot camera takes a picture, it is stored as a black and white image in the robot's memory. Each image is an $H \times W$ grid of pixels, with rows numbered 0 through $H-1$ (0 සිට $H-1$ දක්වා) and columns numbered 0 through $W-1$ (0 සිට $W-1$ දක්වා). There are **exactly two (හරියටම දෙකක්)** black pixels in each image, and all other pixels are white.

The robot can process each image with a program consisting of simple instructions. You are given the values of H, W, and a positive integer K. Your goal is to write a procedure to produce a program for the robot that, for any image, determines whether the **distance** between the two black pixels is exactly K. Here, the distance between a pixel in row r_1 and column c_1 and a pixel in row r_2 and column c_2 is $|r_1 - r_2| + |c_1 - c_2|$. In this formula $|x|$ denotes the absolute value of x, which equals x if $x \geq 0$ and equals $-x$ if $x < 0$.

We now describe how the robot works.

The robot's memory is a sufficiently large array of cells, indexed from 0 (0 සිට අංකනය කරන ලද). Each cell can store either 0 or 1 and its value, once set, will not be changed. [සෑම cell එකකම 0 හෝ 1 ලකුණු කල හැක. එක් වරක් cell එකක ලකුණු කළ පසු, එය වෙනස් කළ නොහැක.] The image is stored row by row in cells indexed 0 through $H \cdot W - 1$. The first row is stored in cells 0 through $W-1$, and the last row is stored in cells $(H-1) \cdot W$ through $H \cdot W - 1$. In particular, if the pixel in row i and column j is black, the value of cell $i \cdot W + j$ is 1, otherwise it is 0.

A robot's program is a sequence of **instructions**, which are numbered with consecutive integers starting from 0 (එනම් 0, 1, 2, 3, 4, ...). When the program is run, the instructions are executed one by one (instructions එකින් එක, පිළිවෙළින් ක්‍රියාත්මක වේ). Each instruction reads the values of one or more cells (we call these values the instruction's **inputs**) and produces a single value equal to 0 or 1 (we call this value the instruction's **output**). The output of instruction i is stored in cell $H \cdot W + i$.
The inputs of instruction i can only be cells that store either pixels or outputs of previous instructions, i.e. cells 0 to $H \cdot W + i - 1$. [ඕනෑම instruction එකකට ලබා දිය හැක්කේ ඒ වනතුරු memory එකෙහි ගබඩා කර ඇති cells පමණි]

There are four types of instructions:

- `NOT`: has exactly one input. Its output is 1 if the input is 0, otherwise its output is 0.
- `AND`: has one or more inputs. Its output is 1 if and only if **all** of the inputs are 1.
- `OR`: has one or more inputs. Its output is 1 if and only if **at least one (අවම වශයෙන් එකක්වත්)** of the inputs is 1.
- `XOR`: has one or more inputs. Its output is 1 if and only if an **odd number** of the inputs are 1.

The output of the last instruction of the program should be 1 if the distance between the two black pixels is exactly K, and 0 otherwise. [කළු pixel දෙක අතර දුර K නම්, අවසාන instruction එකෙහි output එක 1 විය යුතු අතර, K නොවේ නම් output එක 0 විය යුතු වේ]

Implementation details

You should implement the following procedure:

```
void construct_network(int H, int W, int K)
```

- H, W: dimensions of each image taken by the robot's camera
- K: a positive integer
- This procedure should produce a robot's program. For any image taken by the robot's camera, this program should determine whether the distance between the two black pixels in the image is exactly K.

This procedure should call one or more of the following procedures to append instructions to the robot's program (which is initially empty) [robot ගේ මෘදුකාංගයට instructions එකතු කිරීමට පහත procedures, call කළ යුතුයි] :

```
int add_not(int N)
int add_and(int[] Ns)
int add_or(int[] Ns)
int add_xor(int[] Ns)
```

- Append a `NOT`, `AND`, `OR`, or `XOR` instruction, respectively.
- N (for `add_not`): the index of the cell from which the appended `NOT` instruction reads its input
- Ns (for `add_and`, `add_or`, `add_xor`): array containing the indices of the cells from which the appended `AND`, `OR`, or `XOR` instruction reads its inputs
- Each procedure returns the index of the cell that stores the output of the instruction [සෑම procedure එකකින්ම return වන්නේ එයින් එකතු වූ instruction එකෙහි output එක ගබඩා වන cell එකෙහි අංකයයි]. The consecutive calls to these

procedures return consecutive integers starting from $H \cdot W$.

The robot's program can consist of at most $10\,000$ instructions. The instructions can read at most $1\,000\,000$ values in total. In other words, the total length of Ns arrays in all calls to `add_and`, `add_or` and `add_xor` plus the number of calls to `add_not` cannot exceed $1\,000\,000$.

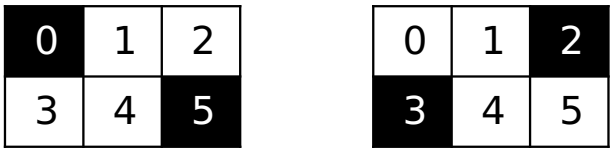
After appending the last instruction, procedure `construct_network` should return. The robot's program will then be evaluated on some number of images. Your solution passes a given test case if for each of these images, the output of the last instruction is 1 if and only if the distance between the two black pixels in the image is equal to K.

The grading of your solution may result in one of the following error messages:

- `Instruction with no inputs`: an empty array was given as the input to `add_and`, `add_or`, or `add_xor`.
- `Invalid index`: an incorrect (possibly negative) cell index was provided as the input to `add_and`, `add_or`, `add_xor`, or `add_not`.
- `Too many instructions`: your procedure attempted to add more than $10\,000$ instructions.
- `Too many inputs`: the instructions read more than $1\,000\,000$ values in total.

Example

Assume $H = 2$, $W = 3$, $K = 3$. There are only two possible images where the distance between the black pixels is 3.



- Case 1: black pixels are 0 and 5
- Case 2: black pixels are 2 and 3

A possible solution is to build a robot's program by making the following calls:

1. `add_and([0, 5])`, which adds an instruction that outputs 1 if and only if the first case holds. The output is stored in cell 6.
2. `add_and([2, 3])`, which adds an instruction that outputs 1 if and only if the second case holds. The output is stored in cell 7.
3. `add_or([6, 7])`, which adds an instruction that outputs 1 if and only if one of the cases above holds.

Constraints

- $1 \leq H \leq 200$
- $1 \leq W \leq 200$
- $2 \leq H \cdot W$
- $1 \leq K \leq H + W - 2$

Subtasks

1. (10 points) $\max(H, W) \leq 3$
2. (11 points) $\max(H, W) \leq 10$
3. (11 points) $\max(H, W) \leq 30$
4. (15 points) $\max(H, W) \leq 100$
5. (12 points) $\min(H, W) = 1$
6. (8 points) Pixel in row 0 and column 0 is black in each image.
7. (14 points) $K = 1$
8. (19 points) No additional constraints.

Sample grader

The sample grader reads the input in the following format:

- line 1: H W K
- line $2 + i$ ($i \geq 0$): $r_1[i]$ $c_1[i]$ $r_2[i]$ $c_2[i]$
- last line: -1

Each line excepting the first and the last line represents an image with two black pixels. We denote the image described in line $2 + i$ by image i. One black pixel is in row $r_1[i]$ and column $c_1[i]$ and the other one in row $r_2[i]$ and column $c_2[i]$.

The sample grader first calls `construct_network(H, W, K)`. If `construct_network` violates some constraint described in the problem statement, the sample grader prints one of the error messages listed at the end of Implementation details section and exits.

Otherwise, the sample grader produces two outputs.

First, the sample grader prints the output of the robot's program in the following format:

- line $1 + i$ ($0 \leq i$): output of the last instruction in the robot's program for image i (1 or 0).

Second, the sample grader writes a file `log.txt` in the current directory in the following format:

- line $1 + i$ ($0 \leq i$): $m[i][0]$ $m[i][1]$ $\ldots$ $m[i][c - 1]$

The sequence on line $1 + i$ describes the values stored in the robot's memory cells after the robot's program is run, given image i as the input. Specifically, $m[i][j]$ gives the value of cell j. Note that the value of c (the length of the sequence) is equal to $H \cdot W$ plus the number of instructions in the robot's program.