















Reap and Sow

You oversee a large plot of land that's task is to provide food for a modest kingdom, Bitland. There are two jobs given to the serfs who help with the harvest. First are the Sowers. Each Sower manages a horizontal parcel of land. No two Sowers' land parcels overlap. These Sowers only plant seeds in their given parcel. Each parcel stretches from one side of the plot of land to the other. See the figure below of a pictorial example of Sowers land distribution.

The second role of the serf is that of a reaper. Each reaper is will harvest the crops from a vertical parcel of land. No two Reapers' land parcels overlap. And each parcel stretches from the top of the plot of land to the bottom.

	Reaper 1	Reaper 2	Reaper 3	Reaper 4	Reaper 5	Reaper 6
Sower 1						
Sower 2						
Sower 3						
Sower 4						
Sower 5						
Sower 6						

For each piece of land there is exactly one reaper and one sower that manages it. Additionally, each intersection of reaper and sower parcel is equal in area to any other intersection. Due to the size of the plot and the resources of the kingdom not every piece of land will be used to grow food. In fact, only half of the land will be used.

To make the serfs' jobs simple either an intersection will be used or unsued. Due to the dirt quality certain parcel intersections might be unusable. Additionally, certain sections of land may already have food growing. The king does not wish to waste food, so you must have the serfs continue to use such sections of land.

There are a few caveats.

1. The serfs wish to work equally. In other words, each reaper will work the same number of parcel intersections, and each sower will work the same number of parcel intersections.
2. The serfs do not wish to be thought of as disposable, so each reaper refuses to have an identical land sequence as another reaper. Also each sower refuses to have an identical land sequence as another sower. However, a sower can work the same sequence as a reaper, because they don't do the same job.

3. Additionally the serfs don't like to be repetitive, so each reaper refuses to reap three contiguous intersections (or not reap three contiguous intersections). Each sower refuses to plant three contiguous intersections (or not plant three contiguous intersections).¹

Your job is to determine based on the initial layout of the parcel intersection, whether a distribution of work can be made to satisfy all the sowers and reapers.

Note 1. The contiguous section must all be in the same row or all in the same column You might find yourself write a program for hiring new serfs soon...

Input Specification

The first line of input will be two positive even integers n and m ($n, m < 17$), representing the number of sowers and reapers, respectively. The input will have n remaining lines, the i -th of which will contain m integers representing the current status of i -th horizontal parcel of land. Of these values, the j -th represents the status of the j -th vertical intersection within the i -th horizontal parcel of land. A value of 0 implies the intersection is unfit for farming. A value of -1 means the piece of land can be used or not. A value of 1 means the piece of land already has crops on it.

Output Specification

There might not be an assignment of crops such that the constraints are all satisfiable. If an assignment is possible output a n by m grid of integers either -1 or 1, where the i -th row's j -th value is 1 when the value is i -th horizontal parcel intersection with the j -th vertical parcel is used to grow crops, and -1 otherwise. Each integer on the same row should be space separated. *If you want to follow the sample output format to improve readability you can.*

In the case where no assignment of crops satisfies all the constraints output the word "impossible" (quotes for clarity).
















There will be at most one solution.

Input Output Example

























Input	Output
6 6 -1 -1 -1 0 -1 -1 1 -1 1 1 -1 -1 0 1 0 -1 -1 -1 0 0 1 -1 1 -1 1 1 -1 -1 -1 -1 0 -1 1 -1 -1 -1	1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1
8 6 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 -1 -1 0 -1 1 1 -1 -1 -1 -1 1	0 1 0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 0 1 0 1

Explanation

Case 1: The below layout corresponds to the first plot

	Reaper 1	Reaper 2	Reaper 3	Reaper 4	Reaper 5	Reaper 6
Sower 1						
Sower 2						
Sower 3						
Sower 4						
Sower 5						
Sower 6						

The plants that satisfy the conditions are in the green squares as follows,

	Reaper 1	Reaper 2	Reaper 3	Reaper 4	Reaper 5	Reaper 6
Sower 1						
Sower 2						
Sower 3						
Sower 4						
Sower 5						
Sower 6						

The white squares are unfarmed.

Grading Information

Reading from and writing to standard input – 10 points

Using recursive backtracking (no I don't want to see your iterative approach) – 10 points

Comments, white space usage, and reasonable variable names – 10 points

No output aside from the answer (e.g. no input prompts) – 10 points.

Your program will be tested on 12 test cases – 5 points each

No points will be awarded to programs that do not compile.

Only cases that finish within the maximum of {5 times my solution, 10 seconds} will be graded.