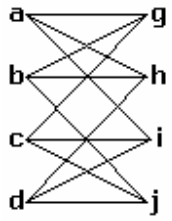
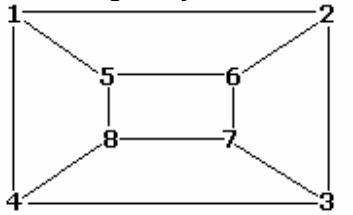
FORBIDDEN SUBGRAPH

Two undirected graphs *G* and *H* are said to be *isomorphic* if:

- they have the same number of vertices and
- a one-to-one correspondence exists between their vertices so that, for any two distinct vertices of *G*, there exists an edge between them if and only if there exists an edge between their corresponding vertices in *H*.

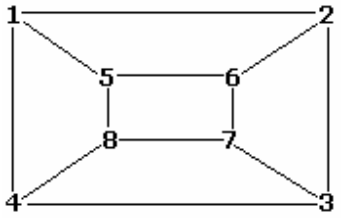
For example, the next two graphs are isomorphic, even though they look different here:

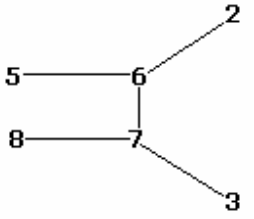




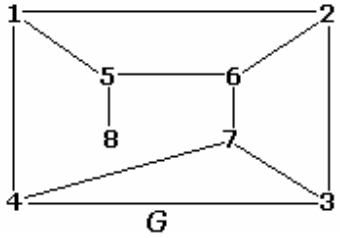
A possible one-to-one correspondence showing that these two graphs are isomorphic is given by {a-1, b-6, c-8, d-3, g-5, h-2, i-4, j-7}, but others exist too.

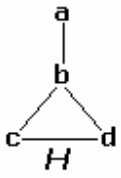
A *subgraph* of a graph *G* is a graph whose sets of vertices and edges are subsets of those in *G*. Note that *G* is a subgraph of itself. The following example shows a graph and one of its subgraphs:

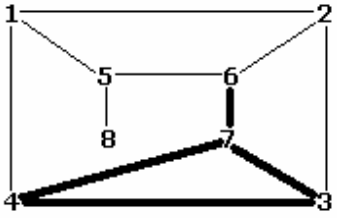




We say that a graph *G contains* another graph *H* if there is at least one subgraph *H'* of *G* which is isomorphic to *H*. The following figure shows a graph *G* that contains the graph *H.*







TASK

Given two undirected graphs *G* and *H,* produce a subgraph *G'* of *G* such that:

- the number of vertices in *G* and *G'* is the same and
- *H is* ***not*** *contained in G'*.

Naturally, there may be many subgraphs G' with the above properties. Produce one of those subgraphs with as many edges as possible.

BASE ALGORITHM

Perhaps the most basic strategy to approach this problem is to consider the edges of *G* in the order that they are represented in the input file, then attempting to add them one by one to *G'*, verifying at each step whether *H* is contained in *G'* or not. The correct implementation of this greedy algorithm will earn some points, but much better strategies exist.

CONSTRAINTS

$3 \le m \le 4$ The number of vertices of *H.*
$3 \le n \le 1000$ The number of vertices of *G.*

INPUT

You will be given 10 files `forbidden1.in` to `forbidden10.in` each with the following data:

`forbiddenK.in`	**DESCRIPTION**

`3 5`
`0 1 0`
`1 0 1`
`0 1 0`
`0 1 0 0 0`
`1 0 1 0 0`
`0 1 0 1 0`
`0 0 1 0 1`
`0 0 0 1 0`
	LINE 1: Contains two space-separated integers, respectively: *m* and *n*.
	NEXT m LINES: Each line contains *m* space-separated integers and represents one vertex of *H* in the order 1, ..., *m*. The *i*-th element of the *j*-th line in this section is equal to 1 if vertices *i* and *j* are joined by an edge in *H* and is equal to 0 otherwise.
	NEXT n LINES: Each line contains *n* space-separated integers and represents one vertex of *G* in the order 1, ..., *n*. The *i*-th element of the *j*-th line in this section is equal to 1 if vertices *i* and *j* are joined by an edge in *G* and is equal to 0 otherwise.

Observe that, except for line 1, the above input represents the adjacency matrices of *H* and *G*.

OUTPUT

You must provide 10 files, one for each of the inputs. Each file must contain the following data:

`forbiddenK.out`	**DESCRIPTION**

`#FILE forbidden K`
`5`
`0 1 0 0 0`
`1 0 0 0 0`
`0 0 0 0 0`
`0 0 0 0 0`
`0 0 0 0 0`
	LINE 1: The file header. The file header must contain **`#FILE forbidden K`** where K is a number between 1 and 10 that corresponds to the input file solved.
	LINE 2: Contains one integer: *n*.
	NEXT n LINES: Each line contains *n* space-separated integers and represents one vertex of *G'* in the order 1, ..., *n*. The *i*-th element of the *j*-th line in this section is equal to 1 if vertices *i* and *j* are joined by an edge in *G'*, and is 0 otherwise.

Observe that, except for lines 1 and 2, the above output represents the adjacency matrix of *G'.* Note that there are many possible outputs, and that the above output is correct but not optimal.

GRADING

Your score will depend on the number of edges in the *G'* you output. Your score will be determined in the following way: you will receive a non-zero score for each output file only if it meets the task specification. If it does, your score will be calculated as follows. Let E_y be the number of edges in your output, let E_b be the number of edges in *G'* as computed by the BASE ALGORITHM, and let E_m be the maximum number of edges in the output of any of the contestants submissions. Your score for the case will be:

- $30\ \boldsymbol{E_y / E_b}$ if $\boldsymbol{E_y \le E_b}$, or
- $30 + 70(\boldsymbol{E_y - E_b})/(\boldsymbol{E_m - E_b})$ if $\boldsymbol{E_y > E_b}$.