Moscow Workshops ICPC МФТИ SBERBANK hello muscat programming bootcamp Phaze\Ventures HARBOUR SPACE UNIVERSITY GUtech

Problem A. Complete graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

A *complete* undirected graph is a graph in which every two distinct vertices are connected with an edge.

How many edges does a complete undirected graph with n vertices contain?

Input

A single integer n ($1 \le n \le 1000$), the number of vertices.

Output

Print a single integer: the number of edges in a complete undirected graph with n vertices.

Examples

standard input	standard output
2	1
3	3
4	6

Scanned by CamScanner

Problem B. Degrees of vertices

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A *degree* of a vertex in an undirected graph is the number of edges adjacent to the vertex.

You are given an adjacency matrix of a graph. For each vertex, print its degree.

Input

In the first line of input there is a single integer n ($1 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

Print n numbers. The i-th of them should equal the degree of vertex i.

Example

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	1 3 2 2

Scanned by CamScanner

Problem C. Path

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes

A *path* between two vertices s and t in an undirected graph is a sequence of vertices $s = v_0, v_1, \ldots, v_{k-1}, v_k = t$ such that there is the graph contains an edge (v_{i-1}, v_i) for each $1 \le i \le k$.

Two vertices are connected if there exists a path between them. In other words, two vertices are connected if one can move from one of them to another using graph edges.

You are given an adjacency matrix of an undirected graph. If there is a path between vertices 1 and n, print "Yes", otherwise print "No".b

Input

In the first line of input there is a single integer n ($2 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

Print "Yes" if there is a path between vertices 1 and n, and "No" otherwise.

Examples

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	Yes
3 0 1 0 1 0 0 0 0 0	No

you have number of vertices and adj matrix.
Do dfs from 1 to n.
If you find it → print yes
else → NO.

Scanned by CamScanner

Problem D. Distances

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes

A *path* between two vertices s and t in an undirected graph is a sequence of vertices $s = v_0, v_1, \ldots, v_{k-1}, v_k = t$ such that there is the graph contains an edge (v_{i-1}, v_i) for each $1 \le i \le k$. The length of the path is the number of edges in it. In particular, the length of the path between any vertex and itself is 0.

The distance from vertex s to vertex t is the length of the shortest path connecting those vertices.

You are given an adjacency matrix of an undirected graph. Find the distance from vertex 1 to all other vertices.

Input

In the first line of input there is a single integer n ($2 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

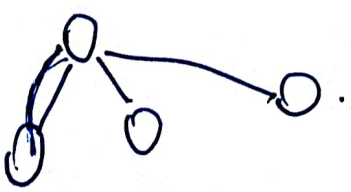
Output

Print n numbers. The i-th of them should be the length of the shortest path from vertex 1 to vertex i or -1 if the path does not exist.

Examples

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	0 1 2 2
3 0 1 0 1 0 0 0 0 0	0 1 -1

Do BFS from 1 to {1, 2, 3 ... n}.
if you find a path, print the length
if not → print -1.



Scanned by CamScanner

Problem E. Triangle

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes

Three vertices x, y, z form a *triangle* in a graph if the graph contains edges (x, y), (x, z) and (y, z).

You are given an undirected graph. Find any three vertices that form a triangle in it or report that there are none.

Input

In the first line of input there is a single integer n ($1 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

If there is a triangle in the graph, print three distinct numbers between 1 and n: the indices of vertices that form the triangle. If there are several answers, print any of them.

If there is no triangle, print a single number -1.

Examples

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	4 3 2
3 0 1 0 1 0 0 0 0 0	-1

for i < n
  for [redacted] j < i
    if has edge (i, j) has edge, has edge.
      find k such that (i, k), (j, k)

Scanned by CamScanner

Problem F. Connected components

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A *path* between two vertices s and t in an undirected graph is a sequence of vertices $s = v_0, v_1, \ldots, v_{k-1}, v_k = t$ such that there is the graph contains an edge (v_{i-1}, v_i) for each $1 \le i \le k$.

Two vertices are connected if there exists a path between them. In other words, two vertices are connected if one can move from one of them to another using graph edges.

A connected component is a subset of vertices of the graph such that:

1. every two vertices from the subset are connected,
2. no other vertex is connected to any vertex in the subset.

For example, the complete graph has one connected component which includes all the vertices, and the empty graph with n vertices has n connected components, each of them being a single vertex.

You are given an adjacency matrix of an undirected graph. Find the number of connected components in it.

Input

In the first line of input there is a single integer n ($2 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

Print a single integer: the number of connected components in the graph.

Examples

standard input	standard output
5 0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0	2
3 0 1 0 1 0 1 0 1 0	1

Scanned by CamScanner

Problem G. Adjacency matrix to edges list

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes

You are given an adjacency matrix of an undirected graph. Print the list of all edges of this graph.

Input

In the first line of input there is a single integer n ($2 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

In the first line print a single integer m, the number of edges in the graph. In each of the next m lines print two numbers, the vertices connected by the corresponding edge.

Edges may be printed in any order, and each edge should be printed exactly once.

Examples

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	4 1 2 2 3 2 4 3 4
3 0 1 0 1 0 0 0 0 0	1 1 2

for i < size
  for j < i
    if graph[i][j].
      vector.push_back(make_pair(i, j)).

cout << vector.size() << endl;
print the elements of the vector.

Scanned by CamScanner

Problem H. Shortest Path

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes

A *path* between two vertices s and t in an undirected graph is a sequence of vertices $s = v_0, v_1, \ldots, v_{k-1}, v_k = t$ such that there is the graph contains an edge (v_{i-1}, v_i) for each $1 \le i \le k$. The length of the path is the number of edges in it. In particular, the length of the path between any vertex and itself is 0.

You are given an adjacency matrix of an undirected graph. Find the shortest path from vertex 1 to vertex n (that is, the path that has minimum possible number of edges).

Input

In the first line of input there is a single integer n ($2 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

If the path between vertices 1 and n does not exist, print -1.

Otherwise print a single integer k, the number of vertices in the path (including 1 and n). Then print k numbers denoting the vertices in order they occur in the path.

If there are multiple answers, print any of them.

Examples

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	3 1 2 4
3 0 1 0 1 0 0 0 0 0	-1

int dist[101][101]; → init to 101

func or define min(a,b)

BFS(a, dist)

for each child of root

Scanned by CamScanner

Problem I. Cycle in Undirected Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

A *cycle* in an undirected graph is a sequence of at least three vertices $v_1, v_2, \ldots, v_k$ such that the graph contains an edge (v_{i-1}, v_i) for each $1 < i \leq k$ and an edge (v_k, v_1). The cycle is *simple* if all vertices in it are distinct.

You are given an adjacency matrix of an undirected graph. Find any cycle in this graph or report that no cycles exist.

Input

In the first line of input there is a single integer n ($2 \leq n \leq 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

If no cycle exists, print -1.

Otherwise print a single integer k, the number of vertices in the cycle. Then print k numbers denoting the vertices in order they occur in the cycle.

If there are multiple answers, print any of them.

Examples

standard input	standard output
4 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0	3 2 3 4
3 0 1 0 1 0 0 0 0 0	-1

ret = {}
For each node (n)
  add n to ret
  Loop through all nodes (n1)
    add node to ret
    get children of n1
    for all children

Scanned by CamScanner

Moscow Workshops ICPC МФТИ SBERBANK hello muscat programming bootcamp Phaze\Ventures HARBOUR.SPACE UNIVERSITY GUtech

Problem J. Topological sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

A *cycle* in an directed graph is a sequence of at least two vertices $v_1, v_2, \ldots, v_k$ such that the graph contains an edge (v_{i-1}, v_i) for each $1 < i \le k$ and an edge (v_k, v_1). The cycle is *simple* if all vertices in it are distinct.

A *topological sort* of a directed graph is an order of its vertices $v_1, \ldots, v_n$ such that for any edge (u, v) vertex u is earlier in the order than v.

You are given an adjacency matrix of an undirected graph. If there is a cycle in the graph, print it. Otherwise print the topological sorting of the graph.

Input

In the first line of input there is a single integer n ($2 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge from vertex i to vertex j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of a directed graph without loops. That is, $e_{i,i} = 0$ for each i. Also the graph does not contain parallel edges (that is, at most one of $e_{i,j}$ and $e_{j,i}$ is 1.

Output

If a cycle exists, print a word "`cycle`" (without quotes). In the next line print a single integer k, the number of vertices in the cycle. Then print k numbers denoting the vertices in order they occur in the cycle.

Otherwise print a word "`topsort`" (without quotes). In the next line print n distinct numbers from 1 to n, the order of vertices in the topological sort order.

If there are multiple cycles or sort orders, print any of them.

Examples

standard input	standard output
4 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0	cycle 3 2 3 4
3 0 0 1 0 0 0 0 1 0	topsort 1 3 2

Scanned by CamScanner

Problem K. Is It Chain?

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 mebibytes

A *chain graph* is a undirected graph such that all its vertices form a chain. Formally, it is possible to reorder the vertices in order $v_1, \ldots, v_n$ such that the graph will contain edges $(v_1, v_2), \ldots, (v_{n-1}, v_n)$ and no other edges.

Given a graph, check if it is a chain graph. If it is the case, print its vertices in the chain order.

Input

In the first line of input there is a single integer n ($1 \le n \le 100$), the number of vertices in the graph.

Next n lines contain the adjacency matrix of the graph. Each of the next n lines contains n integers each. The j-th integer in i-th line is $e_{i,j}$ and equals 1 if there is an edge between vertices i and j, and 0 otherwise.

It is guaranteed that the matrix is an adjacency matrix of an undirected graph without loops. That is, $e_{i,i} = 0$ for each i and $e_{i,j} = e_{j,i}$ for each i, j.

Output

If the graph is a chain, print n distinct numbers from 1 to n, the indices of vertices in the same order in which they occur in the chain.

If there is no chain, print a single number -1.

Examples

standard input	standard output
4 0 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0	1 2 4 3
3 0 1 1 1 0 1 1 1 0	-1

Scanned by CamScanner

Problem L. Shortest Path with Edge List

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

You are given an edge list of an undirected graph. Find the shortest path from vertex 1 to vertex n (that is, the path that has minimum possible number of edges).

Input

In the first line of input there are two integers n, m ($1 \le n, m \le 10^5$), the number of vertices and the number of edges in the graph.

Each of the next m lines contains two integers u_i, v_i. That means that there is an edge (u_i, v_i) in the graph.

It is guaranteed that there are no loops and multiple edges in the graph. That is, $u_i \neq v_i$ for all i and each unordered pair (u_i, v_i) occurs at most once in the input.

Output

If the path between vertices 1 and n does not exist, print -1.

Otherwise print a single integer k, the number of vertices in the path (including 1 and n). Then print k numbers denoting the vertices in order they occur in the path.

If there are multiple answers, print any of them.

Examples

standard input	standard output
4 4 1 2 2 3 3 4 4 2	3 1 2 4
3 1 1 2	-1

Scanned by CamScanner