I. Complete and complement

Difficulty: Hard **Time**: 2 s **Memory**: 1024 MB

by bucketpotato

You are given an undirected **disconnected** graph G_1 with n vertices. You can perform the following two operations on G_1:

1. Pick distinct vertices x, y, z, such that the edges (x, y) and (y, z) are in the graph, but (x, z) is not in the graph. Then, add the edge (x, z) to the graph.
2. For all pairs of distinct vertices x, y: if the edge (x, y) is in the graph, then remove it from the graph. Otherwise, add it to the graph.

Now you are given a second graph G_2 with the same vertex set. Transform G_1 into G_2 using at most n^2 operations or say it's impossible.

Input

Input consists of multiple tests. The first line contains t $(1 \le t \le 250)$, the number of tests.

The first line of each test contains n $(2 \le n \le 10^3)$, the number of vertices in G_1 and G_2 .

The next n lines each contain strings of length n, consisting of "0"s and "1"s, which describe G_1. The j-th character of the i-th string (denoted $s_{i,j}$) is "1" if there is an edge between vertex i and vertex j, and is "0" otherwise.

The next n lines each contain strings of length n, describing G_2 in the same format as G_1.

It is guaranteed that for both graphs, and for all $1 \le i, j \le n$, that $s_{i,j} = s_{j,i}$ and $s_{i,i} = 0$.

It is guaranteed that G_1 is disconnected. It is **NOT** guaranteed that G_2 is disconnected.

It is guaranteed that the sum of n across all tests does not exceed 10^3.

Output

For each test, if the goal is impossible, output -1.

Otherwise, on the first line, output k $(0 \le k \le n^2)$, the number of operations you perform.

On the following k lines: first output an integer $1 \le \text{type} \le 2$ denoting the type of move you wish to perform.

If $\text{type} = 1$, then you should also output 3 distinct integers $1 \le x, y, z \le n$ such that edges (x, y) and (y, z) exist in the graph, but (x, z) does not. **Note that the order in which you output these numbers matters.**

If there are multiple possible answers using at most n^2 moves, you can output any. You do not need to minimize the number of moves. It can be shown that if there exists a way to transform G_1 into G_2, there is a way using at most n^2 moves.

Sample 1

Input

```
2
2
00
00
01
10
2
00
00
00
00
```

Output

```
1
2

2
2
2
```

Explanation

We start with a graph with 2 vertices and no edges.

- In the first test, G_2 is the complement of G_1, so we just need to apply operation 2 once.
- In the second test, G_2 is already equal to G_1, so we don't need to apply any operations. But we don't need to minimize the number of operations, so it's fine if we perform operation 2 twice.

Sample 2

Input

```
2
4
0100
1000
0001
0010
0111
1011
1101
1110
4
0100
1000
0001
0010
0100
1010
0101
0010
```

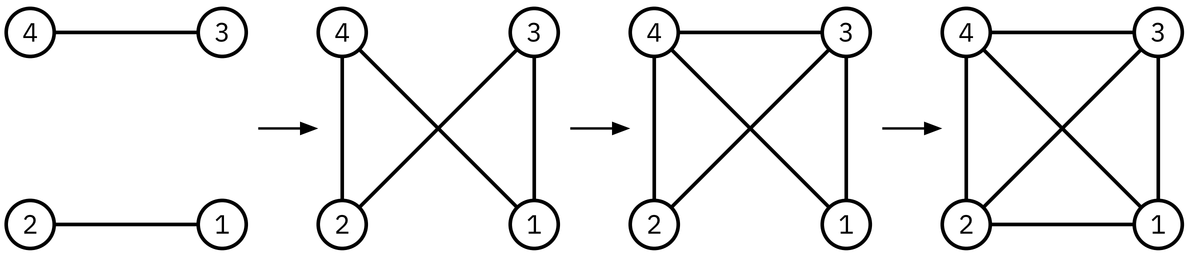
Output

```
3
2
1 4 1 3
1 2 4 1

-1
```

Explanation

In the first test, the sequence of operations performed looks like this:



In the second test, we can show there is no answer.