3845 Garrison

Barracks in a garrison are structured in N rows and N columns with certain security arrangements at each row-column position. Most of the positions are well guarded; however certain positions remain weakly guarded. All positions on the boundary, either well guarded or weakly guarded, are considered vulnerable. With respect to a given vulnerable weakly guarded position, positions appearing next to it column-wise/ row-wise/ diagonally, are considered vulnerable. Thus for a given vulnerable weakly guarded position there are at most eight vulnerable positions. Positions, which are neither vulnerable nor weakly guarded, are considered strongly secured. You are required to write a program to find all strongly secured positions in the garrison.

Let a zero '0' identify a weakly guarded position and a one '1' identify a wellguarded position. For illustration, in a garrison with barracks in 5 rows and 5 columns, the strongly secured positions are identified below by (1).

```
0     1     1     1     0

1     0     1     1     1

1     1     1    (1)    1

1    (1)    0    (1)    1

1     1     1     1     1
```

Input

The input may contain multiple test cases.

For each case, the first line gives the value of N, which is not greater than 20.

The next N lines identify security status of each position in the garrison. The i-th line identifies the status in the i-th row of barracks, $i = 1, 2, \ldots, N$. Each line appears as a string of size N, containing '0's and '1's. Each '0' identifies a weakly guarded position while each '1' identifies a well-guarded position.

A line containing a zero '0' as the first character follows the last test case.

Output

For each test case there are two output lines.

The first line gives the total number S of strongly secured positions. If S is equal to 0 then there is no further output.

Otherwise there is an additional line of output. It identifies all S positions, listing row number and column number of positions in row-major order. The line contains $2 \times S$ integers separated by space.

Sample Input

```
5
01110
10111
11111
11011
```

```
11111
5
01110
10111
11011
11001
11111
4
1111
1011
1101
1111
5
11111
10111
11011
11001
11111
0
```

Sample Output

```
3
3 4 4 2 4 4
0
2
2 3 3 2
5
2 3 2 4 3 2 3 4 4 2
```