6325 Connecting Islands

The civil war in the Bytelandian ocean has ended. The two Bytelandian islands have decided to unite forming the United Kingdom of Bytelandia. One of the first concerns of the newly chosen government is transportation.

Prior to the war each of the 2 islands had a perfect network of transportation. On any given island, every city was connected by a train to every other city. Connections are bidirectional. During the war, some train lines were disrupted and thus, the trains on these lines were suspended. On any island, no more than 15 trains were suspended. In addition to rerunning the disrupted trains, the government has to form inter-island ferry connections. The ferries will connect every city X to every city Y, given that X and Y are not on the same island. Commuting between cities on the same island will only be via trains.

In Bytelandia, everything is binary, including train and ferry ticket prices. A ticket costs either 0 or 1 units of currency. The government decided not to change any ticket prices for trains that were not disrupted during the war. It will assign ticket costs to the trains suspended during the war and to all ferry lines. Priding themselves in being strong mathematicians, Bytelandians have decided to design the ticket system such that for any 3 distinct cities X, Y and Z,

$$C(X,Y) + C(Y,Z) \geq C(X,Z)$$

where $C(X,Y)$ is the ticket price of getting from X to Y (by train if they belong to the same island or by ferry otherwise).

You are given a description of the Bytelandian islands, your goal is to figure out a costs assignment if it is possible.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case is the description of both islands (one after the other). The description of one island starts with one line containing one integer C_k represents the number of cities in this island, followed by C_k lines each one contains C_k characters ($1 \leq C_k \leq 100$). Each character is '`0`', '`1`' or '`x`'. The j-th character of the i-th line is the price of the train ticket from city i to city j on that island. A price '`x`' means that it is one of the suspended train lines. Each C_k by C_k table is guaranteed to be symmetric (table[i][j] = table[j][i] for each different i and j), and each table will have a diagonal of '`0`'s (table[i][i] = '0' for each different i), also each table will contain no more than 30 '`x`'s (15 different trains, because each train will be mentioned twice in the table).

Note that the input might contain 3 different cities in the same islands and the trains connecting them are not suspended, and these 3 cities don't satisfy the described condition above.

Output

For each test case, if there is no way to do the costs assignment satisfying the conditions described above, output on a single line '`NO`' (without the quotes).

Otherwise, print '`YES`' (without the quotes) on the first line followed by $C_1 + C_2$ lines each one contains $C_1 + C_2$ characters C_1 is the number of cities in the first island, and C_2 is the number of cities in the second island). The j-th character on the i-th line is the cost of getting from city i to city j

(by train if they belong to the same island or by ferry otherwise). In the output, the cities on the first island are numbered from 1 to C_1 (inclusive) while the cities in the second island are numbered from $C_1 + 1$ to $C_1 + C_2$ (inclusive). If there is more than one correct solution, print anyone of them.

Note that you can't change the costs given in the input, and the output table must be symmetric.

Sample Input

```
2
3
O1l
lOx
1x0
2
01
10
2
Ox
xO
4
OlOx
1010
0100
x000
```

Sample Output

```
YES
01101
10011
10011
01101
11110
NO
```