In engineering sciences, partial differential equations play an important and central role. For example, the temperature of a metal plate can be expressed as a partial differential equation if the temperature on the boundaries is known. This is called a boundary value problem.

Usually, it is not easy to solve these problems. Analytical solutions exist only in very special cases. But there are some more or less "good" numerical ways to solve boundary value problems.

We now will look at one method which works with finite difference approximations for the derivatives of a function. For this approach, we do not look at an analytical function $u(x)$ but we are only interested in the values of u at a finite set of discrete points $x_i : u_i = u(x_i)$. The distance between two adjacent points, x_i and x_{i+1}, is constant: $h = x_{i+1} - x_i$ (cf. Figure 1).

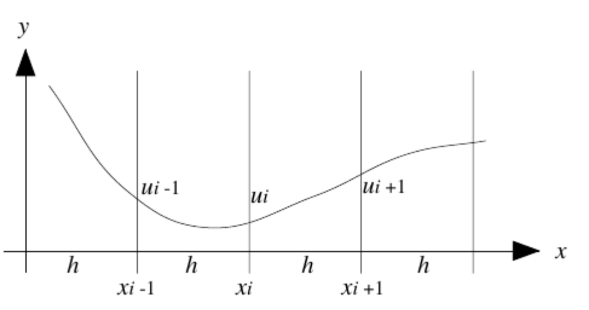

Figure 1: $u(x)$ at some discrete points x_i

The finite difference approximation of a first derivative of the function $u(x)$ is

$$\frac{\partial}{\partial x} u\Big|_i = \frac{1}{2h}(-u_{i-1} + u_{i+1}) \tag{1}$$

The second derivative is approximated by

$$\frac{\partial^2}{\partial x^2} u\Big|_i = \frac{1}{h^2}(u_{i-1} - 2u_i + u_{i+1}) \tag{2}$$

This approximation works with 2-dimensional functions $u(x, y)$ as well. For simplicity we only work on square problems, i.e. (x, y) is element of $[0, 1] \times [0, 1]$. Again, the area of the function is discretized in a similar way: $x_{i+1} - x_i = y_{i+1} - y_i = h = \frac{1}{n}$, for some integer $n \geq 2$. We only look at the values of $u(x, y)$ at the discrete points $P_k = (x_i, y_j) : u_{i,j} = u(P_k)$.

With this discretization, we have a function $u_{i,j}$ as shown in Figure 2:

On the boundary, $u(x_i, y_j)$ is given by 4 known functions:

$$\begin{aligned} u(x_i, 0) &= b_1(x_i) \\ u(x_i, 1) &= b_2(x_i) \\ u(0, y_j) &= b_3(y_j) \\ u(1, y_j) &= b_4(y_j) \end{aligned} \tag{3}$$

The points P_k cover the inner points of the discretization area, i.e. the area without the boundary. They are numbered from left to right and from top to bottom like English text.

What we now want to do is to solve the poisson-equation in the area $[0, 1] \times [0, 1]$:

$$\frac{\partial^2}{\partial x^2} u + \frac{\partial^2}{\partial y^2} u = f(x, y) \tag{4}$$

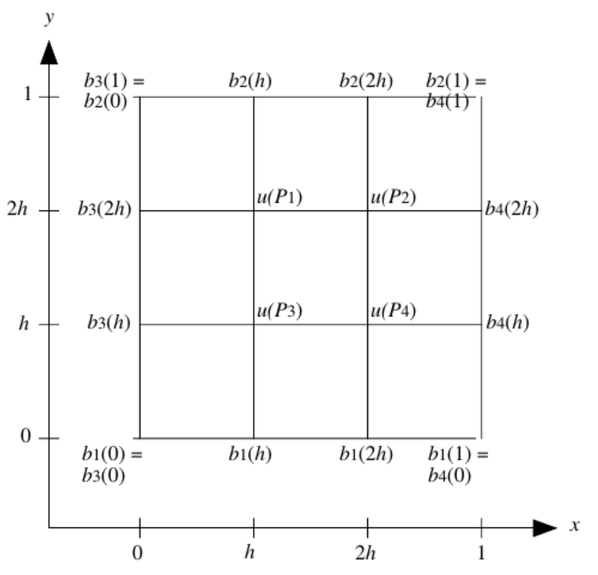

Figure 2 : Function $u_{i,j}$ in the discretization area

with the above boundary conditions.

$f(x, y)$ is a given 2-dimensional function. With equation (2) and the above discretization, the poisson-equation can be approximated at

$$\frac{1}{h^2}(u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1} - 4u_{i,j} = f_{i,j}, \tag{5}$$

where $f_{i,j}$ is the function $f(x, y)$, evaluated at the discrete points (x_i, y_j).

Formula (5) can be written in a more readable form, depending on the position of the discrete points:

$$\frac{1}{h^2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \Diamond u = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \Diamond f \tag{6a}$$

A similar equation, which we will use as an example below, is:

$$\frac{1}{h^2}\begin{pmatrix} 1 & 0 & 2 \\ 0 & -4 & 0 \\ 3 & 0 & 4 \end{pmatrix} \Diamond u = \begin{pmatrix} 0 & 5 & 0 \\ 6 & 0 & 7 \\ 0 & 8 & 0 \end{pmatrix} \Diamond f \tag{6b}$$

We call the 3×3 matrix on the left hand side v and the 3×3 matrix on the right hand side g.

Now, equation (6b) can be formulated in every point of the discrete area of Figure 2:

$$\left.\begin{array}{ll} P_1 : & \frac{1}{h^2}(b_2(0) + 2b_2(2h) - 4P_1 + 3b_3(h) + 4P_4) = 5f(h, 1) + 6f(0, 2h) + 7f(2h, 2h) + 8f(h, h) \\ P_2 : & \frac{1}{h^2}(b_2(h) + 2b_2(1) - 4P_2 + 3P_3 + 4b_4(h)) = 5f(2h, 1) + 6f(h, 2h) + 7f(1, 2h) + 8f(2h, h) \\ P_3 : & \frac{1}{h^2}(b_3(2h) + 2P_2 - 4P_3 + 3b_1(0) + 4b_1(2h)) = 5f(h, 2h) + 6f(0, h) + 7f(2h, h) + 8f(h, 0) \\ P_4 : & \frac{1}{h^2}(P_1 + 2b_4(2h) - 4P_4 + 3b_1(h) + 4b_1(1)) = 5f(2h, 2h) + 6f(h, h) + 7f(1, h) + 8f(2h, 0) \end{array}\right\} \tag{7}$$

and (7) is a linear equation system for the values of $u(x, y)$ at the points P_1, P_2, P_3 and P_4.

By rearranging and adding the terms on each line, the linear equation system can be formulated as:

$$az = b \tag{8}$$

where a is a 4×4 matrix and b is a vector with 4 elements. Vector z represents the unknown values of $u(x, y)$ at the points P_1, P_2, P_3 and P_4.

You are to write a program that creates the linear equation system (7) in the form (8) for any two matrices v and g (6). As input, the two matrices v and g and the functions b_1, b_2, b_3, b_4 , and f are given. Also, a parameter n is given as the number of discretization intervals. Thus, $h = \frac{1}{n}$. As the result, your program should calculate the matrix a and the vector b. For this more general case, there are $(n - 1)^2$ inner points and a and b must be sized accordingly.

Input

The input file consists of m tests. The number m is given in the first line of the file. The first line of each test contains the number n which gives the number of discretizations intervals as defined above. You may assume that $2 \leq n \leq 30$. Then the 3×3 matrices v and g follow. The following four lines contain the functions b_1, b_2, b_3 and b_4, each given as a vector of order $n + 1$, containing the values for 0, h, $2h$, ..., 1. Finally, the function f is given as a $n + 1$ by $n + 1$ matrix. Like the vectors before, it contains the values for $x, y = 0, h, 2h, \ldots, 1$. Each row contains from left to right the function values for increasing x values while each column contains from top to bottom the function values for decreasing y values.

A vector occupies one line. Its values are given in ascending order, separated by a space. A n by n matrix occupies n lines. Its rows are given in ascending order as vectors, which occupy one line each. All values found in the input file are integer values.

Output

For each test found in the input file, your program should output the matrices a and b. Matrix a is a $(n - 1)^2 \times (n - 1)^2$ matrix (the discretization area (cf. Figure 2) contains $(n - 1)^2$ inner points, which are unknown). The vector b is of order $(n - 1)^2$. They should be output in the same format as the vectors and matrices in the input file. Your output should only contain integer values. Note that the expression $\frac{1}{h^2}$ yields an integer number and that all other calculations can also be done using integer numbers.

Sample Input

```
1
3
1 0 2
0 -4 0
3 0 4
0 5 0
6 0 7
0 8 0
3 4 5 6
0 1 2 3
3 2 1 0
6 5 4 3
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
```

Sample Output

```
-36 0 0 36
0 -36 27 0
0 18 -36 0
9 0 0 -36
-35 -188 -189 -315
```