Matrix Normalization

You are given two vectors $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, and an $n \times n$ matrix $\mathbf{A}$.

Let $\mathbf{D}$ be an $n \times n$ diagonal matrix with diagonal elements from vector $\mathbf{d}$:

$$\mathbf{D} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix}$$

Let $\mathbf{V}$ be an $n \times n$ diagonal matrix with diagonal elements from vector $\mathbf{v}$:

$$\mathbf{V} = \begin{pmatrix} v_1 & 0 & \cdots & 0 \\ 0 & v_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & v_n \end{pmatrix}$$

Your task is to **efficiently** compute the matrix product $\mathbf{DAV}^T$.

Input Format

- First, 1d numpy array of size n containing integers $d_1, d_2, \ldots, d_n$
- Second, 1d numpy array of size n containing integers $v_1, v_2, \ldots, v_n$
- Third, 2d numpy array of size $n \times n$ containing integers representing matrix $\mathbf{A}$

Output Format

Output the resulting $n \times n$ matrix $\mathbf{DAV}^T$. Each element should be returned as an integer.

Constraints

- $\mathbf{d} \in \mathbb{R}^n$
- $\mathbf{v} \in \mathbb{R}^n$
- $\mathbf{A} \in \mathbb{R}^{n \times n}$
- $1 \leq n \leq 1000$
- All input values are integers in the range $[-1000, 1000]$

Sample Input

```
d = [2 3 1]
v = [1 2 4]
A = [[1 2 3],
    [4 5 6],
    [7 8 9]]
```

Sample Output

```
[[2 8 24],
 [12 30 72],
 [7 16 36]]
```

Implementation

Goal: Fill in the following function:

```
def matrix_norm(d, v, A):
    ...
    return ... # Return the resulting matrix
exec("\n".join(iter(input, "#Exit"))) # Don't remove this line
```