Decomposition

Given J vectors $x_k \in \mathbb{R}^n$, for $k = 1, 2, \ldots, J$, stored in the matrix $X \in \mathbb{R}^{n \times J}$ as columns, and given a vector $c = (c_1, c_2, \ldots, c_J) \in \mathbb{R}^J$, compute

$$A = \sum_{i=1}^{J} c_i x_i x_i^T$$

The matrix X has the following structure:

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,J} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,J} \end{pmatrix} = \begin{pmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_J \\ | & | & & | \end{pmatrix}$$

where each column x_k is an n-dimensional vector.
Your task is to **efficiently** compute the matrix A which is the weighted sum of outer products.

Input Format

- First, 2d numpy array X of size $n \times J$ containing integers
- Second, 1d numpy array c of size J containing integers $c_1, c_2, \ldots, c_J$

Output Format

Output the resulting $n \times n$ matrix $A = \sum_{i=1}^{J} c_i x_i x_i^T$. Each element should be returned as an integer.

Constraints

- $X \in \mathbb{R}^{n \times J}$
- $c \in \mathbb{R}^J$
- $1 \leq n \leq 500$
- $1 \leq J \leq 500$
- All input values are integers in the range $[-100, 100]$

Sample Input

```
X = [[1, 2],
     [3, 4],
     [5, 6]]
c = [1, 2]
```

Sample Output

```
[[ 9  19  29],
    [19  41  63],
    [29  63  97]]
```

Implementation

Goal: Fill in the following function:

```
def decomposition(X, c):
     return ... # Return the resulting matrix
exec("\n".join(iter(input, "#Exit"))) # Don't remove this line
```

Hint

- numpy broadcasting (`https://numpy.org/doc/stable/user/basics.broadcasting.html`)