Matrix Multiplication

A pxq matrix of real numbers can be represented by a p-slot list, each slot containing a list of size q,

for example,

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{bmatrix}$$

Converted to

```
[ [1, 2, 3, 0],
  [2, 3, 0, 1],
  [4, 1, 2, 2] ]
```

Your task is to implemented functions

- **`mult_c(c, A)`** : returns a resulting matrix from multiplying c (a real number) with matrix A.
- **`mult(A, B)`**: returns A x B (this is a matrix multiplication).

```
def read_matrix():
    m = []
    nrows = int(input())
    for k in range(nrows):
        x = input().split()
        r = []
        for e in x:
            r.append( float(e) )
        m.append(r)
    return m

def mult_c(c, A):



def mult(A, B):



exec(input().strip())      # This command is necessary to grade your answer
```

From wikipedia https://en.wikipedia.org/wiki/Matrix_multiplication

If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times p$ matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

the *matrix product* $\mathbf{C} = \mathbf{AB}$ (denoted without multiplication signs or dots) is defined to be the $m \times p$ matrix[5][6][7][8]

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj},$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, p$.

Input

Python commands.

Output

Result from executing the input commands.

Example

Input (from keyboard)	Output (on screen)

`A=read_matrix();print(mul_c(0.5,A))`
`3`
`1 2`
`2 3`
`3 2`
	`[[0.5, 1.0], [1.0, 1.5], [1.5, 1.0]]`

`A=read_matrix();B=read_matrix();print(mul(A,B))`
`3`
`1 2 3`
`1 1 1`
`2 2 2`
`3`
`1 2`
`2 3`
`3 2`
	`[[14.0, 14.0], [6.0, 7.0], [12.0, 14.0]]`