Z algebry liniowej wiemy, że **macierz** to prostokątna tablica liczb. Oraz że możemy ją utożsamiać z pewnym przekształceniem liniowym pomiędzy dwiema przestrzeniami liniowymi. Na takich tablicach możemy wykonywać różne operacje. Waszym zadaniem będzie implementacja dwóch operacji mnożenia.

W kodzie taką macierz będziemy możemy reprezentować jako listę list (tupli) i tak np

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix}$$

to w pythonie

```
[
    [x_11, x_12, .., x_1n],
    [x_21, x_22, .., x_2n],
    .
    .
    [x_d1, x_d2, .., x_dn]
]
```

Aby ładnie się pracowało możecie użyć funkcji wypisującej listę list w macierzowej formie na konsolę

```python
def mprint(m):
    for row in m:
        for i in range(len(row)):
            if i == 0:
                print("|{:^4}".format(row[i]), end="")
            elif i == len(row) -1:
                print("{:^4}|".format(row[i]))
            else:
                print(" {:^4}".format(row[i]), end="")
    print()
```

Zadanie I (1 pkt)

Transformacje pomiędzy przestrzeniami możemy oczywiście składać. Niech X, Y, Z będą przestrzeniami liniowymi oraz T, S przekształceniami liniowymi między nimi

$$T : X \to Y$$
$$S : Y \to Z$$

wtedy istnieje złożenie

$$S \circ T : X \to Z$$

Mnożenie macierzy[1] A, B jest tak zdefiniowane aby macierz iloczynu odpowiadał przekształceniu opisującemu złożeniu operatorów reprezentowanych przez macierze A, B. Oczywiście jeśli typy (wymiary)[2] przestrzeni się nie zgadzają to nie istnieje złożenie więc i nie możemy wykonać mnożenia.
Dla macierzy

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

ich iloczyn jest macierzą $\mathbf{C} = \mathbf{AB}$

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

o współczynnikach

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

Zadanie oczywiście polega na implementacji mnożenia macierzy wczytanych z plików `A.matrix` `B.matrix`. Pamiętajcie, że funkcja powinna rzucać wyjątek (lub jakoś informować o błędzie) jeśli operacji nie można wykonać.

```python
A = [[1,2,3],[4,5,6]] # to wczytaj z pliku A.matrix
B = [[1,2,3,0], [4,5,6,0], [7, 8, 9,0]]# to wczytaj z pliku B.matrix

def mult(a, b):
    ...

mprint(mult(A, B))
mprint(mult(B, A)) # error
```

Dla takich argumentów funkcja `mult` powinna zwrócić `[[30, 36, 42, 0],[66, 81, 96, 0]]` a w drugim przypadku sygnalizować błąd np w postaci wyjątku ”Exception: Matrix dimensions mismatch”

Zadanie II (1 pkt)

Mając przestrzenie liniowe X, Y nad jakimś ciałem liczb (np $\mathbb{R}$) oraz dwa operatory liniowe

$$T : X \to X'$$
$$S : Y \to Y'$$

[1] `https://en.wikipedia.org/wiki/Matrix_multiplication`

[2] każda przestrzeń liniowa skończeniewymiarowa nad ciałem $\mathbb{C}$ ($\mathbb{R}$ itd) jest izomorficzna z $\mathbb{C}^n$ ($\mathbb{R}^n$ itd) gdzie n wymiar przestrzeni.

możemy stworzyć tak zwany iloczyn tensorowy dwóch przestrzeni $X \otimes Y$. Intuicyjnie jest to taka przestrzeń liniowa w której wektory $x \in X$ indeksujemy wektorami $y \in Y$ a nie elementami ciała $\mathbb{R}$. Dodatkowo na takiej przestrzeni możemy też zdefiniować operator liniowy jako tensor działający w oczywisty sposób

$$(T \otimes S)(x \otimes y) = (Tx) \otimes (Sy)$$

W ogólności taki operator wymaga od nas abyśmy zdefiniowali go na bazach przestrzeni X, Y, ale w interesującym nas przypadku skończeniewymiarowym (czyli gdy działamy na macierzach) zgadza się to z tak zwanym mnożeniem Kroneckera[3]. Przykład chyba mówi więcej niż 1000 słów

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \otimes \begin{bmatrix}0 & 5\\6 & 7\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}0 & 5\\6 & 7\end{bmatrix} & 2\begin{bmatrix}0 & 5\\6 & 7\end{bmatrix}\\3\begin{bmatrix}0 & 5\\6 & 7\end{bmatrix} & 4\begin{bmatrix}0 & 5\\6 & 7\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1\times 0 & 1\times 5 & 2\times 0 & 2\times 5\\1\times 6 & 1\times 7 & 2\times 6 & 2\times 7\\3\times 0 & 3\times 5 & 4\times 0 & 4\times 5\\3\times 6 & 3\times 7 & 4\times 6 & 4\times 7\end{bmatrix} = \begin{bmatrix}0 & 5 & 0 & 10\\6 & 7 & 12 & 14\\0 & 15 & 0 & 20\\18 & 21 & 24 & 28\end{bmatrix}$$

Zadanie oczywiście polega na implementacji mnożenia Kroneckera. Musicie wczytać macierze z pliku `A.matrix` i `B.matrix` zaimplementować funkcję `kronecker(A,B)` zwracającą reprezentację $A \otimes B$.

```python
A = [[1,2,3],[4,5,6]] # to wczytaj z pliku A.matrix
B = [[1,2,3,0], [4,5,6,0], [7, 8, 9,0]]# to wczytaj z pliku B.matrix

def kronecker(a, b):
    ...

mprint(kronecker(A, B))
mprint(kronecker(B, A))
```

Wynikiem takiego wywołania powinno być

```
| 1    2    3    0    2    4    6    0    3    6    9    0   |
| 4    5    6    0    8    10   12   0    12   15   18   0   |
| 7    8    9    0    14   16   18   0    21   24   27   0   |
| 4    8    12   0    5    10   15   0    6    12   18   0   |
| 16   20   24   0    20   25   30   0    24   30   36   0   |
| 28   32   36   0    35   40   45   0    42   48   54   0   |

| 1    2    3    2    4    6    3    6    9    0    0    0   |
| 4    5    6    8    10   12   12   15   18   0    0    0   |
| 4    8    12   5    10   15   6    12   18   0    0    0   |
| 16   20   24   20   25   30   24   30   36   0    0    0   |
| 7    14   21   8    16   24   9    18   27   0    0    0   |
| 28   35   42   32   40   48   36   45   54   0    0    0   |
```

[3] `https://en.wikipedia.org/wiki/Kronecker_product`