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 jednej z takich operacji. W kodzie taką macierz będziemy możemy reprezentować jako listę list (tupel 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]
]
```

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

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

[2] `https://pl.wikipedia.org/wiki/Mnoenie_macierzy`

```
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):
    ...

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

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

Dla takich argumentów funkcja `mult` powinna zwrócić `[[30, 36, 42, 0],[66, 81, 96, 0]]` oraz jakiś błąd "Exception: Matrix dimensions mismatch"

Wersja II (0.5 pkt)

Tu zadanie jest podobne. Musicie wczytać macierz z pliku `A.matrix` i zaimplementować funkcję dokonującą jej transpozycji[3] czyli zamianie kolumn z wierszami.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
A = [[1,2,3],[4,5,6]] # to wczytaj z pliku A.matrix

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

def transpose(a):
    ...

mprint(transpose(A))
```

Wynik transpozycji macierzy `A = [[1,2,3],[4,5,6]]` to oczywiście `[[1,4],[2,5],[3,6]]`

[3] Matematyczna interpretacja takiej operacji wymaga trochę szerszego aparatu pojęciowego. Zainteresowani niech wygooglują hasło "przestrzeń dualna" (albo zapiszą się na kurs analizy funkcjonalnej w budynku obok).