1. Дефинирайте клас за **вектор** от геометрията. От тук нататък когато в условието се спомене **вектор** се има в предвид този клас.

- има атрибут цяло число с неговия размер N. Може да се чете, но не и да се пише отвън
- има едномерен динамичен масив от цели числа. Не може да се достъпва отвън
- има единствен конструктор, който получава аргументи размер N и масив. Заделя памет за N на брой елементи и ги чете последователно от подадения масив
- дефинирайте оператор [] за достъпване на елемент от масива по индекс

2. Дефинирайте клас за матрица

- има атрибути цели числа за нейния размер - N за брой редове и M за брой колони
- има едномерен динамичен масив от вектори - всеки вектор представлява една **колона**
- има единствен скрит конструктор с аргументи за размерите. Конструкторът заделя нужната памет за елементите в масива
- има статичен метод `Matrix from_array(int N, int M, int arr)`, който връща инициализирана инстанция на матрица
- има статичен метод `Matrix from_matrix(Matrix other)`, който връща инициализирана инстанция на матрица
- дефинирайте оператор за събиране **+**
    - с число - всеки елемент на матрицата се събира с подаденото число

```
| 1 2 3 |       | 2 3 4 |
| 4 5 6 | + 1 = | 5 6 7 |
```

    - с матрица - всеки елемент на първата матрица се събира със съответния елемент на втората

```
| 1 2 3 |   | 7 8 9 |   | 8 10 12 |
| 4 5 6 | + | 1 2 3 | = | 5  7  9 |
```

- дефинирайте оператор за изваждане **-**
    - с число - от всеки елемент на матрицата се изважда подаденото число

```
| 1 2 3 |       | 0 1 2 |
| 4 5 6 | - 1 = | 3 4 5 |
```

    - с матрица - от всеки елемент на първата матрица се вади съответният елемент елемент на втората

```
| 1 2 3 |   | 7 8 9 |   | -6 -6 -6 |
| 4 5 6 | - | 1 2 3 | = |  3  3  3 |
```

- дефинирайте оператор за умножение *
    - с число - всеки елемент на матрицата се умножава с подаденото число

```
| 1 2 3 |       | 2 4 6 |
| 4 5 6 | * 2 = | 8 10 12 |
```

    - с матрица - всеки елемент от резултантната матрица представлява сбор от произведенията на съответни елементи в ред на първата и колона на втората матрица. [картинка (https://miro.medium.com/max/1200/1*YGcMQSr0ge_DGn96WnEkZw.png)](https://miro.medium.com/max/1200/1*YGcMQSr0ge_DGn96WnEkZw.png)

```
| 1 2 3 |   | 3 |   | 3*1 + 3*2 + 3*3 |   |  18 |
| 4 5 6 | * | 7 | = | 7*4 + 7*5 + 7*6 | = | 105 |
```

    *За умножение е нужно броят **колони** на **първата** матрица да е равен на броя **редове** на **втората** матрица*

- дефинирайте оператор за изход **<<**, който извежда елементите на матрицата във формат

```
| v11 v12 ... v1m |
| v21 v22 ... v2m |
| ...             |
| vn1 vn2 ... vnm |
```

- дефинирайте оператор за достъпване по индекс [], който връща връща векторът на подадения индекс. *my_matrix[3][7] трябва да върне елемента в колона с индекс 3 и ред с индекс 7*
- всички горни методи и оператори да хвърлят изключения при несъвпадащи или нулеви и отрицателни размери

3. Допълнителни:

- имайте минимален брой извиквания на конструктури и копирания на данни при извикване на методи и оператори
- динамичната памет да се почиства и освобождава
- да се обработват хвърляните изключения
- демонстрация на всичко в `main`
- имплементирайте метод транспониране на матрица - завъртане така, че да се разменят редовете и колоните

```
| 1 2 3 |     | 1 4 |
| 4 5 6 | => | 2 5 |
              | 3 6 |
```