1 A Toy MATLAB

需要提交的文件：*matrix.h*。

Overview

> MATLAB 是美国 MathWorks 公司出品的商业数学软件，和 Mathematica、Maple 并称为三大数学软件。它在数学类科技应用软件中在数值计算方面首屈一指，广泛应用于矩阵运算、绘制函数和数据、实现算法、创建用户界面、连接其他编程语言的程序等。

矩阵是 MATLAB 中的基本元素。MATLAB 用由一对 `[]` 括起的若干数字表示一个矩阵：

```
[ 1 2 3 ; 4 5 6 ; 7 8 9 ]
```

其中行内元素以 `,` 或空格隔开，行与行间以 `;` 分隔。
你的任务是参照 MATLAB，实现一个带有基本矩阵运算功能的矩阵类。
简单来说，你需要实现一个带模板参数 `T` 的类 `Matrix`。

```
template <typename T> class Matrix;
```

表示一类元素类型为 `T` 的矩阵。具体地，它需要支持以下特性：

Features

含参构造

```
int seq[4] = {1, 2, 3, 4};
Matrix<int> a(2, 2, seq); // a := [ 1 2 ; 3 4 ]
Matrix<int> b(1, 2); // b := [ 0 0 ]
```

使用含参构造函数声明一个**已初始化**的矩阵实例。
第一个参数 `h` 表示矩阵的高（行数），第二个参数 `w` 表示矩阵的宽（列数）。**保证** $h, w > 0$ **且** $h \cdot w \leq 10^6$。
第三个参数为初始化序列，序列的第 $i \times w + j$（**0-base**）项表示矩阵第 `i`（**0-base**）行第 `j`（**0-base**）列的元素的初始值。
初始化序列**可省缺**，省缺时表示用元素类型**默认值**（即 `T()`）填充矩阵。

拷贝构造

```
Matrix<int> c(a); // c := [ 1 2 ; 3 4 ]
```

使用拷贝构造函数复制一个相同类型矩阵实例。
矩阵类型 `T` 不匹配时拷贝构造是**非法的**（保证不会在测试中出现）。

矩阵赋值

```
b = b; // b := [ 0 0 ]
c = b; // c := [ 0 0 ]
```

使用赋值运算将等号右端的对象值赋给等式左端对象。不同形状矩阵间的赋值是**合法的**。
矩阵类型 `T` 不匹配时赋值运算是**非法的**（保证不会在测试中出现）。

元素访问

```
Matrix<int> tmp1(a); // tmp1 := [ 1 2 ; 3 4 ]
const Matrix<int> tmp2(a); // tmp2 := [ 1 2 ; 3 4 ]

int ans1 = tmp2(1, 2); // ans1 := 2
tmp1(2, 2) = 1; // tmp1 := [ 1 2 ; 3 1 ]
```

使用 **1-base** 下标访问矩阵中的某一元素，对于非常量矩阵该访问应**可写**。
当访问的下标非法（例如 `ans1(0, 2)` 或 `ans1(1, 3)`）时，应立即停止运算，抛出 `MatrixIndexingError` 类型异常。

矩阵加法

```
Matrix<int> tmp3(1, 4, seq); // tmp3 := [ 1 2 3 4 ]
Matrix<int> tmp4(4, 1, seq); // tmp4 := [ 1 ; 2 ; 3 ; 4 ]

Matrix<int> ans2 = a + a; // ans2 := [ 2 4 ; 6 8 ]
Matrix<int> ans3 = tmp3 + tmp4;
// ans3 := [ 2 3 4 5 ; 3 4 5 6 ; 4 5 6 7 ; 5 6 7 8 ], implicit expansion
```

对于**形状相同**的矩阵，矩阵加法定义为

$$C_{n\times m} := A_{n\times m} + B_{n\times m} \iff c_{ij} = a_{ij} + b_{ij} \quad (0 \le i < n, 0 \le j < m)$$

特别地，对于**行向量 + 列向量**（`[1 x n] + [m x 1]` 或 `[m x 1] + [1 x n]`，这里 `[a x b]` 表示一个 `a` 行 `b` 列的矩阵）型矩阵加法，尽管矩阵形状不同，可以通过分别在行、列方向上重复补齐为 `[m x n]` 的矩阵，使矩阵加法有意义。例如：

$$\begin{bmatrix} a & b & c \end{bmatrix} + \begin{bmatrix} d \\ e \end{bmatrix} \Rightarrow \begin{bmatrix} a & b & c \\ a & b & c \end{bmatrix} + \begin{bmatrix} d & d & d \\ e & e & e \end{bmatrix} = \begin{bmatrix} a+d & b+d & c+d \\ a+e & b+e & c+e \end{bmatrix}$$

这一特性在 MATLAB 中被称为 **Implicit Expansion**。
若矩阵形状不相同且不满足上述 Implicit Expansion 条件，应立即停止运算，并抛出 `MatrixSizeError` 异常。
参与运算的矩阵类型不匹配时加法运算是**非法的**（保证不会在测试中出现）。

矩阵乘法

```
Matrix<int> ans4 = a * a; // ans4 := [ 7 10 ; 15 22 ]
Matrix<int> ans5 = tmp3 * tmp4; // ans5 := [ 30 ]
```

矩阵乘法定义为

$$C_{n\times s} := A_{n\times m} \times B_{m\times s} \iff c_{ij} = \sum_{k=0}^{m-1} a_{ik} \times b_{kj} \quad (0 \le i < n, 0 \le j < s)$$

当相乘的两个矩阵对应维大小不匹配时，应立即停止运算，抛出 `MatrixSizeError` 类型异常。
参与运算的矩阵类型 `T` 不匹配时乘法运算是**非法的**（保证不会在测试中出现）。

关于模板参数

保证类型 `T` 有定义默认构造函数、拷贝构造函数、析构函数、`operator=`、`operator==`、`operator+`（加法）、`operator*`（乘法）。
保证在正确使用类型 `T` 时不会抛出异常、内存泄漏、产生非法或未定义行为。

Structure

请补全以下代码实现 `Matrix` 类。
你可以选择自己喜欢的方式来存储矩阵元素值信息，例如：

- 使用二维数组指针：`T** mat`
- 使用嵌套的 `std::vector` 容器：`std::vector<std::vector<T>> mat`
- 将矩阵元素排列为一维序列，并使用一维数组指针 `T* arr` 或 `std::vector<T> arr` 存储
- 其他任何你喜欢的方式

特别地，以下存储方式不被接受：

- 使用固定大小的一维或二维数组
- 任何将不可避免地导致内存泄漏的实现

```
#ifndef MATRIX_H_
#define MATRIX_H_

#include <exception>
// just include whatever you want

namespace sjtu {

class MatrixSizeError : public std::exception {
```

```
 public:
  const char* what() noexcept {
    return "matrix size mismatch";
  }
};
class MatrixIndexingError : public std::exception {
 public:
  const char* what() noexcept {
    return "invalid matrix indexing";
  }
};

template <typename T>
class Matrix {
 private:
  int h, w;
  // TODO: choose your way to store elements in the matrix

 public:
  int height() const {
    return h;
  }
  int width() const {
    return w;
  }

  Matrix() = delete;
  Matrix(int h_, int w_, T* seq = nullptr) {
    // TODO
  }
  Matrix(const Matrix& o) {
    // TODO
  }

  Matrix& operator= (const Matrix& rhs) {
    // TODO
  }

  T& operator() (int row, int col) {
    // TODO
  }
```

```
  const T& operator() (int row, int col) const {
    // TODO
  }

  Matrix operator* (const Matrix& rhs) const {
    // TODO
  }

  Matrix operator+ (const Matrix& rhs) const {
    // TODO
  }

  ~Matrix() {
    // TODO
  }
};

} // namespace sjtu

#endif // MATRIX_H_
```

Samples

见下发数据包。

Testcases

本题对矩阵运算效率无硬性要求，部分测试点有内存泄漏检查。
保证运算过程中产生的所有矩阵大小（$h \cdot w$）不超过 10^6。

测试点编号	主要考查内容	内存泄漏检查	分值
1	样例	无	5
2	构造析构	无	10
3	矩阵赋值、元素访问	无	10
4	矩阵运算（不含 Implicit Expansion、异常处理、const）	无	10
5	矩阵运算（含 Implicit Expansion、异常处理、const）	无	15
6	鲁棒性测试	无	10
7	综合测试	无	20
8–14	1–7 号测试点	有	20