01矩阵

我们需要实现一个矩阵类，支持基本的矩阵运算和操作，但我们只实现01矩阵。

以下矩阵的运算除了是模2运算以外与一般的矩阵没有区别，
熟悉矩阵和模k运算的同学可以跳过这部分介绍

有关矩阵的基础介绍可以看百度百科。

而**01矩阵**则指，矩阵的每个元素只能是数字 `0` 或者 `1` ；比如这样：

```
A = [0 0 1 1   B = [1 0   C = [0   D = [1]
     1 1 0 0         1 1]       1
     0 1 0 1]                   0]
```

所需实现的内容

我们的01矩阵上应该实现这些数学运算：

- 转置：使用T()函数实现
- 加法：运算符+实现
- 减法：运算符-实现
- Hadamard Product/对应元素之积：运算符*实现
- 矩阵乘法/点积：运算符%实现

以及这些操作：

- 以某个固定模式生成数据以填充矩阵（后文会给出该固定模式的描述）
- 根据二维索引修改某个矩阵元素的值
- 使用cout输出矩阵

数学运算的具体定义

符号书写约定：矩阵表示为大写字母，如A, B, C；矩阵的转置表示形如A^T，B^T，C^T，用$(A)_{ij}$表示矩阵A在第i行j列的值

转置就是按主对角线翻转：

$$B_{ij} = A_{ij}^T = A_{ji}$$

可见显然有 `B[i][j] = A[j][i]`

加法就是行列号相同的对应元素相加：

$$(A - B)_{ij} = A_{ij} + B_{ij}$$

这是加法的计算表格：
（左上角为运算符，第一行表示第一个操作数的取值，第一列表示第二个操作数的取值，其余行列为相应的运算结果）

+	**0**	**1**
0	0	1
1	1	0

注意：01矩阵上定义的加法**不会**产生进位，因此严格意义上是模二加法，也称为异或。

减法就是行列号相同的对应元素相减：

$$(A + B)_{ij} = A_{ij} - B_{ij}$$

思考：模二运算下，减法和加法的计算表格实际上是相同的，为什么？

标量乘法很简单，即行列号相同的对应元素相乘：

$$(A * B)_{ij} = A_{ij} * B_{ij}$$

这是标量乘法的计算表格：

*	0	1
0	0	0
1	0	1

思考：01矩阵上定义的乘法又可以视作哪个逻辑连接词的真值表？

矩阵乘法就是**模二运算**下的普通矩阵乘法：
（我们用百分号%来表示点积的运算符，假设A为$n \times q$的矩阵，B为$q \times m$的矩阵）

$$(A\%B)_{ij} = \sum_{k=1}^{q} A_{ik} * B_{kj}$$

求和符号Σ的展开形式是这样，此处的+运算就是之前定义的+运算：

$$\sum_{k=1}^{n} x_k = x_1 + x_2 + ... + x_n$$

其他操作的定义

1. 初始化数据填充

本题中使用两种方式生成测试用的矩阵初始数据，对应需在构造函数中加入一个参数seed：

- seed = 0时，矩阵元素以010为循环节，一直填满所有元素的位置（先按顺序填第一行，第一行填满后填第二行，直到填满）
- seed = 1时，矩阵元素以110为循环节，一直填满所有元素的位置

如填充方式0产生的3行5列的矩阵 `Matrix(0, 3, 5)` 对应如下：

```
01001
00100
10010
```

如填充方式1产生的2行4列的矩阵 `Matrix(1, 2, 4)` 对应如下矩阵：

```
1101
1011
```

2 修改元素值

使用[]下标获取或改变元素值，如$A[i][j] = a$;

3 打印矩阵

即向cout输出，每行后有换行，列间没有空格，注意最后一行也需要换行。

样例 & 输出

类设计框架如下：

```
class Matrix {
private:
  int* elements;
  // 注意：由于本题主要考察重载的实现，同时也考察内存的动态申请和释放
  //       所以矩阵元素的存储必须使用指针+分配内存的形式，不得使用stl容器
public:
  Matrix(int seed, int row, int column);
  // 其余函数自行设计
  // 要注意：所有数学运算都不能是就地的(inplace)，
  //         即不能改变原矩阵，而应当返回一个新矩阵
}
```

你的接口必须能够实现以下调用（请逆推出来w）：

```
#include "Matrix.h"
using namespace std;
int main() {
  Matrix A(0, 3, 5);
  cout << A << endl;
  Matrix B = A;
  cout << B << endl;
  Matrix C(0, 5, 4);
  Matrix D = A % C;
  cout << D << endl;
  A = D = B;
  Matrix E(1, 5, 3);
  cout << B % E << endl;
  A[0][3] = 1;
```

```
  cout << A << endl;
  B = A.T();
  cout << B << endl;
  cout << Matrix(1, 2, 4) << endl;
  cout << (Matrix(1, 2, 4)%Matrix(0, 4, 3)) + Matrix(0, 2, 3) << endl;
  cout << (Matrix(0, 2, 2) - Matrix(1, 2, 2)) << endl;
  cout << (Matrix(0, 2, 2) * Matrix(1, 2, 2)) << endl;
}
```

我们保证所有的测试输入都合法，不存在两个参与运算的矩阵大小不匹配的情况。

上例的正确输出应当为：

```
01001
00100
10010

01001
00100
10010

0000
0010
0000

000
110
000

01011
00100
10010

001
100
010
101
100

1101
1011

000
000

10
```

```
01

01
00
```

可能有一些不符合C++直觉的设定

比如在赋值的时候一个10×10的矩阵可能需要赋值给一个变量A，但是这个A本来存储的矩阵为2×3的。

```
Matrix A(0, 10, 10);
Matrix B(0, 2, 3);
B = A;
```

首先这是符合正常的书写习惯的，用户不需要记住哪个变量之前存的矩阵是什么形状。其次，在进行了这种赋值运算后，B本来存储的空间被释放，B这个符号得到了重新利用。

返回值而非引用（A=B后，对B的修改不应该影响A），也是一个例子。本题所要实现的功能总体上说是符合直觉可理解的，这也是所有程序库设计的一个准则。

代码提交

代码提交与前几次一样，相信各位同学已经熟悉了。
由于此次基本没有预先规定好的接口形式（运算符重载也有不止一种写法，所以不作规定），不提供.h文件。

提交的应为一个zip压缩包，压缩包内包含以下两个文件，不应该有任何文件夹
Matrix.cpp 包含所有函数的实现
Matrix.h 包含所有函数和类的声明

对题目以及oj存在任何问题可以在课程大群或者助教群提问，我们会尽快回复。

关于编译环境

oj环境如下
gcc version 4.9.4 (Ubuntu 4.9.4-2ubuntu1~14.04.1)

oj在评测的时候，将使用以下命令编译

```
g++ -std=c++11 Matrix.cpp main.cpp -o main
```

main.cpp中包含测试用例，学生不需要提交