Problem 1. 阶乘

描述

输入一个正整数 N，请输出其阶乘（1×2×...×N）。

输入格式

输入共 1 行，一个正整数 N。

输出格式

输出一个整数，即 N 的阶乘。

样例

输入

```
5
```

输出

```
120
```

数据规模和范围

1 <= N <= 20

Problem 2. 螺旋矩阵

描述

一个 N 行 N 列的螺旋矩阵可由如下方法生成：

从矩阵的左上角（第 1 行第 1 列）出发，初始时向右移动；如果前方是未曾经过的格子， 则继续前进，否则右转；重复上述操作直至经过矩阵中所有格子。根据经过顺序，在格子中依次填入 1, 2, 3, ... , N^2（N 的平方），便构成了一个螺旋矩阵。

现给出矩阵大小 N，请你输出这个螺旋矩阵。

输入格式

输入共 1 行，一个整数 N。

输出格式

输出共 N 行，每行 N 个整数，代表螺旋矩阵各行各列的值。

样例

输入

```
5
```

输出

```
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
```

数据规模和范围

1 <= N <= 100

Problem 3. 小明的防火墙

描述

小明开设了一个个人博客网站。有时候他需要阻止某些恶意 IP 地址访问他的网站，但又要允许其他某些 IP 地址（如他的小伙伴们）能访问他的网站。为了达成这个目的，他在配置文件中写了 N 条规则，就像这样：

```
allow 1.2.3.4/30
deny 1.1.1.1/32
allow 127.0.0.1/32
allow 123.234.12.23/3
deny 0.0.0.0/0
```

每条规则格式都是 `[allow|deny] [address]/[mask]`：

- `allow` 代表允许访问请求，`deny` 代表阻止访问请求。
- `address` 代表 IP 地址，范围是 `0.0.0.0` 到 `255.255.255.255`。
- `mask` 代表 IP 地址的子网掩码，即匹配 IP 地址时需要满足的二进制位数，范围是 `0` 到 `32` 的整数。

IP 地址小数点分割的四部分可以分别写为 8 位二进制数（不足则补零），将它们从左向右依次连接起来可以形成一个 32 位二进制数，这就是 IP 地址的二进制形式。如 `1.2.3.4` 四部分的二进制分别是 `00000001`、`00000010`、`00000011`、`00000100`，连接起来的二进制数 `00000001000000100000001100000100` 就是 IP 地址 `1.2.3.4` 的二进制形式。

当有一个 IP 要访问网站时，需要从规则列表中第一条开始依次尝试匹配。若匹配成功，则根据规则决定访问被允许（allow）还是阻止（deny）。如果没有一条规则能匹配，则访问被允许（allow）。其中，若 IP 和规则地址的前 `mask` 个二进制位是一致的，会匹配成功，否则会匹配失败。

例如，IP `128.127.8.125` 可以匹配规则 `deny 128.127.4.100/20`，原因是二进制数 **10000000011111110000**010001100100（`128.127.8.125` 的二进制形式）的前 20 位（由 mask 给定）与二进制数 **10000000011111110000**100001111101（`128.127.4.100` 的二进制形式）的前 20 位是一致的。

现在给定 N 条规则和 M 个访问网站的 IP 地址，你的任务是根据规则找出哪些访问是允许的，哪些访问是需要被阻止的。

输入格式

第 1 行：两个整数 N 和 M。

第 2 到 N+1 行：每行一条规则。

第 N+2 到 N+M-1 行：每行一个访问网站的 IP 地址。

输出格式

对于每个访问请求，依次输出 `YES` 或者 `NO`（注意大小写），代表访问是允许或被阻止。一共输出 M 行。

样例

输入

```
5 5
allow 1.2.3.4/30
deny 1.1.1.1/32
allow 127.0.0.1/32
allow 123.234.12.23/3
deny 0.0.0.0/0
1.2.3.4
1.2.3.5
1.1.1.1
100.100.100.100
219.142.53.100
```

输出

```
YES
YES
NO
YES
NO
```

解释

- `1.2.3.4` 可以匹配规则 `allow 1.2.3.4/30`，因此输出 `YES`；
- `1.2.3.5` 可以匹配规则 `allow 1.2.3.4/30`，因此输出 `YES`；
- `1.1.1.1` 不可以匹配规则 `allow 1.2.3.4/30`，但可以匹配规则 `deny 1.1.1.1/32`，因此输出 `NO`；
- `100.100.100.100` 不能匹配任何规则直到可以匹配到 `allow 123.234.12.23/3`，因此输出 `YES`；
- `219.142.53.100` 不能匹配任何规则直到可以匹配到 `deny 0.0.0.0/0`，因此输出 `NO`（该规则的 mask = 0，不需要任何长度的前缀一致就匹配，因此无论 IP 是什么都可以匹配上）

数据规模和范围

1 <= N, M <= 1000

附加题: 最佳路径

本题是附加题，分值 20。

描述

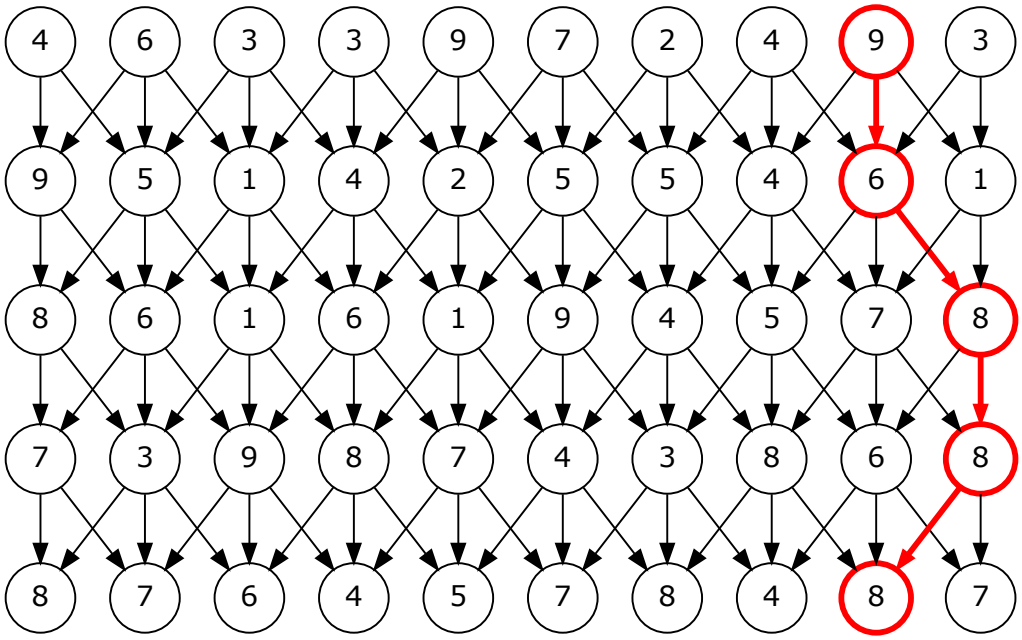
对于一个 R 行 C 列的矩阵，从矩阵顶部到底部有很多条不同的路径。对于每条路径，把路径上面的数加起来可以得到一个和，和最大的路径称为最佳路径（最佳路径可能有多个）。你的任务就是对于给出的矩阵，求出最佳路径上的数字之和。

- 每一步可沿左斜线向下一行、右斜线向下一行或垂直向下一行;
- 矩阵中的数字都是整数，且在区间 `[0, 99]` 中

例如，对于以下这个 5 行 10 列矩阵:

```
4 6 3 3 9 7 2 4 9 3
9 5 1 4 2 5 5 4 6 1
8 6 1 6 1 9 4 5 7 8
7 3 9 8 7 4 3 8 6 8
8 7 6 4 5 7 8 4 8 7
```

如图所示示箭头指出的都是路径上的路，其中标红的是最佳路径，这条路径上面面数字之和是 39。



输入格式

第 1 行：两个整数 R 和 C，代表矩阵行数和列数。

第 1+X 行（1 <= X <= R）：每行 C 个整数，代表矩阵第 X 行上各元素的值。

输出格式

输出共 1 行，一个整数，即所求的最佳路径上的数字之和。

样例

输入

```
5 10
4 6 3 3 9 7 2 4 9 3
9 5 1 4 2 5 5 4 6 1
8 6 1 6 1 9 4 5 7 8
7 3 9 8 7 4 3 8 6 8
8 7 6 4 5 7 8 4 8 7
```

输出

```
39
```

数据规模和范围

1 <= R, C <= 100