数独（sudoku）

【题目描述】

数独是一个有趣的游戏。你需要在一个 9×9 的矩阵中的每个格子中填入 $1\sim9$ 的数字，使得没有两个相同的数字填入同一行、同一列或同一个九宫格中。

整个矩阵被划分为 9 个九宫格，若两个格子同时在最左三列、最右三列或中间三列，且同时在最左三行、最右三行或中间三行，则这两个格子在同一九宫格中。

如果两个相同的数同行、同列或同九宫格，则构成一对冲突。如下列状态中，两个 1 在同一行中，两个 2 在同一列中，两个 3 在同一九宫格中，分别是三对冲突；但两个 4 不是一对冲突。

2	1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	0	3	0
2	0	0	0	0	0	0	0	3

现在有一个数独的初始状态，出题人想对其进行一些修改和询问操作。需要注意：在操作时，初始状态中的数也可以被删除或者合并时被替换。

1. 向目前状态中的指定位置填入一个数：但有可能这个位置已经有一个数了，此时你需要输出一行 **Error!**，然后不进行这次修改操作；在指定的这个位置没有数的情况下，这个数已经与之前存在的在同一行、列或九宫格中的数构成冲突，此时，你需要按照行、列、九宫格的顺序，找到第一种冲突的情况，输出一行 **Error:row!**，**Error:column!** 或 **Error:square!**，然后不进行这次修改操作；否则，你需要输出 **OK!**，并在指定位置填入该数。
2. 删除目前状态中的一个位置上的数：若这个位置没有数字，此时你需要输出一行 **Error!**，然后不进行任何操作；否则你需要输出一行 **OK!**，并将该位置的数删除。
3. 查询目前状态中的某个位置能填入多少种数字；若被查询的位置已经有数字了，你需要输出一行 **Error!**；否则，输出一行一个整数 n 表示能填入的数字个数，随后 n 行每行一个整数，按照从小到大的顺序输出能填入的数字。
4. 将之前的第 i 次操作后的数独状态和第 j 次操作后的数独状态进行合并，作为当前状态。需要注意：对于所有的 5 种操作，包括但不限于出现 **Error!** 或是没有进行任何修改，均被算作一次操作。合并时以行为第一关键字，列为第二关键字的顺序依次考虑每个格子，若第 i 次操作后的数独状态中该位置有数且不会与之前冲突则优先填入；否则，在不会与之前冲突的情况下，填入第 j 次操作后的数

独状态中该位置的数。若均没有数字或均与本次合并中已填入的数字冲突，则不填入任何数。输出一行，包含空格隔开的两个整数，表示最终的结果中有多少数字来自第 i 次操作后的数独状态中，多少来自第 j 次操作后的数独状态中。

5. 查询整个数独的状态，你需要使用方阵格式将整个数独目前的状态输出。方阵格式是一个 19×19 的二维字符数组，具体格式如下，其中用 0 表示该位置还未填入数字。

```
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|5|7|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|7|1|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|9|8|7|6|5|4|3|2|1|
+-+-+-+-+-+-+-+-+-+
```

【输入格式】

从标准输入读入数据。

输入的前 19 行为一个二维字符数组，为数独的初始状态的方阵格式。

随后一行一个整数 T 表示操作的次数。

随后 T 行，每行为下列形式：

Insert x y k，表示在 (x, y) 位置插入数 k。

Delete x y，表示删除 (x, y) 位置的数。

Query x y，表示查询 (x, y) 位置能填入且不会出现冲突的数。

Merge i j，表示合并第 i 次操作后的状态和第 j 次操作后的状态。

Print，表示查询整个数独的状态。

其中 x 表示行数，从上到下分别为 1 到 9，y 表示列数，从左到右分别为 1 到 9。

【输出格式】

输出到标准输出。
对于每个操作，你需要按照题目描述进行对应的输出。

【样例 1 输入】

```
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|5|7|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|7|1|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|9|8|7|6|5|4|3|2|1|
+-+-+-+-+-+-+-+-+-+
9
Insert 7 1 2
Insert 8 1 2
Insert 8 2 2
Query 8 9
Delete 6 1
Delete 8 1
Insert 1 1 1
Merge 6 4
Print
```

【样例 1 输出】

```
OK!
Error:column!
Error:square!
6
4
5
6
7
8
9
OK!
Error!
OK!
13 1
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|5|7|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|2|0|0|0|7|1|0|0|0|
+-+-+-+-+-+-+-+-+-+
|0|0|0|0|0|0|0|0|0|
+-+-+-+-+-+-+-+-+-+
|9|8|7|6|5|4|3|2|1|
+-+-+-+-+-+-+-+-+-+
```

【样例 2】

见题目目录下的 ***2.in*** 与 ***2.ans***。

【子任务】

所有测试点的数据规模与约定如下：

测试点	约定 1	约定 2	约定 3
1	否	否	否
2		是	
3			
4	是	否	
5			
6		是	
7			
8		否	是
9		是	
10			

约定 1：存在插入和删除操作。

约定 2：存在查询单个格子的操作。

约定 3：存在合并操作。

对于所有的数据，$1 \le T \le 100$，$1 \le x, y, k \le 9$，对于第 a 个操作，若是 **Merge** 操作，则 $1 \le i, j < a$。保证第一个操作不是 **Merge** 操作。

对于所有的数据，均可能存在查询整个数独的操作，且保证初始状态不存在冲突。