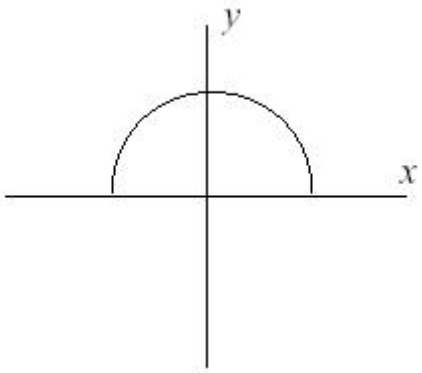
# NOIP 普及组复赛
# 冲刺赛（9 天）

# Problem 1

弗雷德想买地盖房养老，但是土地每年会被密西西比河淹掉一部分，而且经调查是以半圆形的方式淹没的，每年淹没 50 平方英里，以初始水岸线为 x 轴，平分半圆为 y 轴，建立如下图的坐标系




给出坐标点（y>0），让你判断在那一年这个坐标点会被淹没。

**Input**

输入的第一行是一个正整数，表示多少数据集将包含 （N）。

每一条 N 线将包含弗莱德正在考虑的土地的 x 和 y 直角坐标。这些将浮点数以英里。y 坐标是非负坐标。并且不会有（0,0）坐标输入。

**Output**

对于每个数据集，应显示一行输出。 这一行应采取以下形式：

Property N: This property will begin eroding in year Z.

其中 N 是数据集（从 1 开始计数），Z 是第一年（从 1 开始），该属性表示 AT THE END OF YEAR Z。Z 必须是整数。在最后一个数据集之后，应该打印出 ENDOFOUTPUT.

注意：没有财产会出现在半圆形边界上：它将在内部或外部。

这个问题会自动判断。 您的答案必须完全匹配，包括大小写，标点符号和空格。 这包括线路末端的周期。所有地点以英里为单位。

**Sample Input**

2

1.0 1.0

25.0 0.0

**Sample Output**

Property 1: This property will begin eroding in year 1.

Property 2: This property will begin eroding in year 20.

END OF OUTPUT.

# Problem 2

一台计算机：

1)有 M 个 cpu，N 个内存

2)存在一个足够大的工作队列

3)一个工作 Job Ji 需要 Ai 个 cpu 和 Bi 个内存，到达队列的时间为 Ti，完成截止时间为 Ui，完成后的奖励为 Vi。提早完成，额外奖励 Wi 每小时。延迟完成，处罚 Xi 每小时。

4)当一个工作在执行的时候，所使用的 cpu 和内存不能再分配给其它的工作，直到该工作计算完成。

5)每个工作的执行时间为 1 小时。

6)没有其它工作需要执行的时候，空闲，直到有新的工作到达队列。

7)如果有多个工作的到达时间相同，总是先检测价值高的工作，可以认为每个工作的价值都不相同。

8)工作到达时间早的先检测。收入计算：截止时间大于 f 且还没执行的工作不予考虑。截止时间小于等于 f 的工作，完成的收入为工作的价值加上相关的奖励或者处罚，没执行完成的只计算处罚。

**Input**

该输入包含几个测试用例，每个测试用例都描述了大型机的资源和作业列表。 每个测试用例以包含单个整数 F，0 <= F <= 10000，时间线的行开始。 以下行由三个整数 M，N 和 L 组成（M，N，L> = 0）。 M 是主机中 CPU 的数量，N 是存储器大小。 L 表示作业列表中的作业数。 最多将有 10000 个工作。

测试用例中的后续 L 行描述了作业的信息。 描述作业 Ji 的数据由 7 个整数 Ai，Bi，Ti，Ui，Vi，Wi，Xi 组成。 Ai 和 Bi 表示对 CPU 和内存的要求（Ai，Bi> = 0）。 Ti 和 Ui 表示作业的到达时间和时间轴（0 <= Ti <= Ui）。 Vi，Wi，Xi 是工作的奖励，奖金和惩罚（Vi，Wi，Xi> = 0）。输入文件以空测试用例（F = 0）结束。 而且这种情况不应该被处理。

**Output**

您的程序必须根据作业列表计算主机的总收入。 对于每个测试用例，打印案例编号，冒号和空格，然后打印收入。在每个测试用例之后打印一个空行。

注意：不要计算未执行的工作，而且其时间晚于 F.

**Sample Input**

10

4 256 3

1 16 2 3 10 5 6

2 128 2 4 30 10 5

2 128 2 4 20 10 5

0

**Output for the Sample Input**

Case 1: 74

# Problem 3

哈特正在玩一个有趣的游戏，Gnome Tetravex，这些天。 在游戏中，开始时，玩家被给予 n * n 个正方形。 每个正方形分为四个三角形，四个数字（范围从 0 到 9）。 在正方形中，三角形是左三角形，顶三角形，直角三角形和底三角形。 例如，图 1 示出了 2×2 正方形的初始状态。

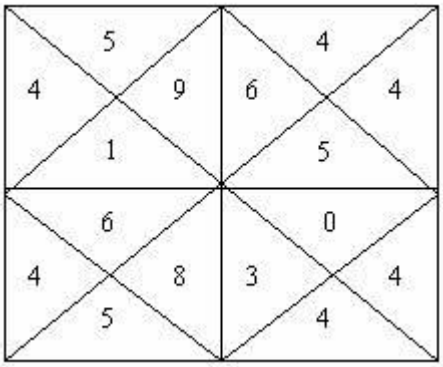



Fig. 1 The initial state with 2*2 squares

玩家需要将正方形移动到终止状态。 在终止状态下，任何两个邻接的正方形应使相邻三角形标有相同的数字。 图 2 示出了上述示例的终止状态之一。

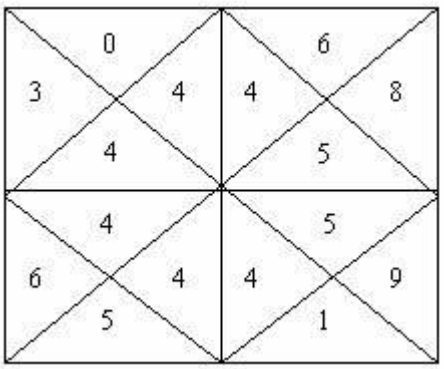



Fig. 2 One termination state of the above example

看来游戏并不那么难。 但事实上，哈特在比赛中还没有完成。 他可以成功完成最简单的游戏。 当面对更复杂的游戏时，他无法找到出路。

有一天哈特正在玩一个非常复杂的游戏，他喊道：“TMD，这是不可能解决的。 对于这样一个可怜的玩家来说，帮助他的最好方法就是告诉他游戏是否可以解决。 如果他被告知游戏是无法解决的，他不必浪费太多的时间。

**Input**

输入文件由几个游戏案例组成。 每个游戏案例的第一行包含一个整数 n，0 <= n <= 5，表示游戏的大小。以下 n * n 行描述了这些三角形的标记号。 每行由四个整数组成，它们代表一个正方形的顶三角形，右角三角形，底三角形和左三角形。在最后一个游戏的情况下，整数 0 表示输入数据集的终止。

**Output**

你应该决定游戏案例是否可以解决。 对于每个游戏的情况，打印游戏编号，冒号和空格，然后显示您的判断。 如果游戏是可解的，打印字符串“可能”。 否则，请打印“不可能”表示无法解决问题。在每个游戏案例之间打印一个空白行。

注意：任何不需要的空白行或白色空格都是不可接受的。

**Sample Input**

2

5 9 1 4

4 4 5 6

6 8 5 4

0 4 4 3

2

1 1 1 1

2 2 2 2

3 3 3 3

4 4 4 4

0

**Output for the Sample Input**

Game 1: Possible

Game 2: Impossible

# Problem 4

我们要预测一些关于并行运行两个程序的单个处理器的行为的事实。 程序是根据以下语法的命令序列：

<Program> → <Command> *

<Command> → <Variable> := <Operand> <Operator> <Operand>

<Operator> → + | -

<Operand> → <Variable> | <Constant>

A<Variable>是以一个字母（不区分大小写）开始的（至多 20 个）字母数字字符（A ... Z，a ... z 和 0 ... 9）的序列。 A <constant>是无符号整数（小于 100）。 令牌之间可能有任意数量的空白或制表符。在执行之前，程序被翻译成机器语言。 X：= Y + Z 形式的语句转换为以下机器指令集：

Mov R1, Y

Mov R2, Z

Add R1, R2

Mov X, R1

MOV 指令将其第二个操作数的内容复制到其第一个操作数中。 一个 Add（Sub）指令从其第一个操作数中加上（减）其第二个操作数，结果存储在第一个操作数中。 请注意，Y 和 Z 表示变量或整数常数。 对于命令 X：= Y-Z 生成的指令与上述指令相似，除了使用 Sub 命令而不是 Add。

处理器被给予两个机器语言程序，并从第一个指令开始执行它们。 在每个步骤中，它随机选择两个程序中的一个，并从所选程序运行下一条指令。 这一直延续到一个程序结束。 在这种情况下，来自另一个的剩余指令被顺序地执行到结束并且处理器停止。假设所有变量在两个程序之间共享，但每个程序都有一个单独的寄存器集。 该程序的目标是计算所有可能执行程序中所有变量的预期最终值。 更准确地说，我们要考虑两

个程序的每一个可能的执行，并且对于每个变量，计算不同执行中其最终值的平均值。假设所有变量的初始值为零。

**Input**

输入的第一行包含单个整数 t（1 <= t <= 10），测试用例数，后跟每个测试用例的输入数据。 测试用例由一条空白行分开。 每个测试用例由一对程序组成。 每个程序被写成一连串的连续行，每行都包含一个命令。 程序以只包含单词 END 的行结束。 您可以假设任何程序中的变量都不被命名为“END”。一个测试用例程序之间没有空白行。每个程序中至少有一行，最多 25 行。 两个程序中的变量总数不超过 10 个。

**Output**

对于每个测试用例，输出应包含所有变量的预期最终值，按变量名称的字母顺序排列（以此顺序排列的字母数字）。 不同测试用例的输出应该分开一个空白行。 将输出中的数字舍入小数点后的 4 位数。 不要在小数点后面省略零尾（例如写入 1.2000 而不是 1.2）。

**Sample Input**

1

S := 1 + 3

END

S := S+S

END

**Sample Output**

3.0000