2006 年百度之星程序设计大赛初赛题目

变态的比赛规则

为了促进各部门员工的交流，百度 (baidu) 举办了一场全公司范围内的 " 拳皇友谊赛 " ，负责组织这场比赛的是百度的超级 " 拳皇 " 迷 W.Z. W.Z 不想用传统的淘汰赛或者循环赛的方式，而是自己制定了一个比赛规则。

由于一些员工（比如同部门或者相临部门员工）平时接触的机会比较多，为了促进不同部门之间的交流， W.Z 希望员工自己组成不同组。不同组之间的每两个人都会进行一场友谊赛而同一组内的人则之间不会打任何比赛。

比如 4 个人，编号为 1--4, 如果分为两个组并且 1,2 一个组， 3 ， 4 一个组，那么一共需要打四场比赛： 1 vs 3,1 vs 4,2 vs 3,2 vs 4. 而如果是 1,2,3 一组， 4 单独一组，那么一共需要打三场比赛 : 1 vs 4,2 vs 4,3 vs 4.

很快 W.Z 意识到，这样的比赛规则可能会让比赛的场数非常多。 W.Z 想知道如果有 N 个人 , 通过上面这种比赛规则，总比赛场数有可能为 K 场吗？比如 3 个人，如果只分到一组则不需要比赛，如果分到两组则需要 2 场比赛 , 如果分为三组则需要 3 场比赛。但是无论怎么分都不可能只需要 1 场比赛。

相信作为编程高手的你一定知道该怎么回答这个问题了吧？ 那么现在请你帮助 W.Z 吧。

输入

每行为一组数据，包含两个数字 N, K 。 (0<N<=500, K>=0)

输出

对输入的 N,K 如果 N 个员工通过一定的分组方式可能会一共需要 K 场比赛，则输出 "YES", 否则输出 "NO", 每组数据占一行。

所有的输入输出均为标准输入输出。

例子

输入文件 :

2 0

2 1

3 1

3 2

输出 :

YES

YES

NO

YES

2006 年百度之星程序设计大赛初赛题目 4

2007-05-14 17:39

剪刀石头布

N 个小孩正在和你玩一种剪刀石头布游戏。 N 个小孩中有一个是裁判，其余小孩分成三组（不排除某些组没有任何成员的可能性），但是你不知道谁是裁判，也不知道小孩们的分组情况。然后，小孩们开始玩剪刀石头布游戏，一共玩 M 次，每次任意选择两个小孩进行一轮，你会被告知结果，即两个小孩的胜负情况，然而你不会得知小孩具体出的是剪刀、石头

还是布。已知各组的小孩分别只会出一种手势（因而同一组的两个小孩总会是和局），而裁判则每次都会随便选择出一种手势，因此没有人会知道裁判到底会出什么。请你在 M 次剪刀石头布游戏结束后，猜猜谁是裁判。如果你能猜出谁是裁判，请说明最早在第几次游戏结束后你就能够确定谁是裁判。

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为两个整数 N 和 M（ $1 \le N \le 500$ ， $0 \le M \le 2000$ ），分别为小孩的个数和剪刀石头布游戏进行的次数。接下来 M 行，每行两个整数且中间以一个符号隔开。两个整数分别为进行游戏的两个小孩各自的编号，为小于 N 的非负整数。符号的可能值为“＝”、“＞”和“＜”，分别表示和局、第一个小孩胜和第二个小孩胜三种情况。

输出格式：

每组测试数据输出一行，若能猜出谁是裁判，则输出身为裁判的小孩的编号，并输出在第几次游戏结束后就能够确定谁是裁判。如果无法确定谁是裁判，或者发现剪刀石头布游戏的胜负情况不合理（即无论谁是裁判都会出现矛盾），则输出相应的信息。具体输出格式请参考输出样例。

输入样例：

```
3 3
0<1
1<2
2<0
3 5
0<1
0>1
1<2
1>2
0<2
4 4
0<1
0>1
2<3
2>3
1 0
```

输出样例：

```
Can not determine
Player 1 can be determined to be the judge after 4 lines
Impossible
Player 0 can be determined to be the judge after 0 lines
```

说明：

共有 5 个测试数据集，每个测试数据集为一个输入文件，包含多组测试数据。每个测试数据集从易到难分别为 5 、 10 、 15 、 30 和 40 分，对每个测试数据集分别执行一次程序，每次必须在运行时限 3 秒内结束程序并输出正确的答案才能得分。

所有数据均从标准输入设备（ stdin/cin ）读入，并写出到标准输出设备 （ stdout/cout ）中。

五个测试数据集中输入 N 分别不大于 20 、 50 、 100 、 200 和 500 ，各有 10 组测

试数据。

2006 年百度之星程序设计大赛初赛题目 5
座位调整
题目描述：
百度办公区里到处摆放着各种各样的零食。百度人力资源部的调研发现，员工如果可以在自己喜欢的美食旁边工作，工作效率会大大提高。因此，百度决定进行一次员工座位的大调整。调整的方法如下：
1 ． 首先将办公区按照各种零食的摆放分成 N 个不同的区域。（例如：可乐区，饼干区，牛奶区等等）。
2 ．每个员工对不同的零食区域有不同的喜好程度（喜好程度度的范围为 1 — 100 的整数，喜好程度越大表示该员工越希望被调整到相应的零食区域）。
3 ． 由于每个零食区域可以容纳的员工数量有限，人力资源部希望找到一个最优的调整方案令到总的喜好程度最大。
数据输入：
第一行包含两个整数 N ， M ，（ 1<=N ， M<=300 ）。分别表示 N 个区域和 M 个员工。
第二行是 N 个整数构成的数列 a ，其中 a[i] 表示第 i 个区域可以容纳的员工数，(1<=a[i]<=M ， a[1]+a[2]+..+a[N]=M) 。
紧接着是一个 M*N 的矩阵 P ， P （ i ， j ）表示第 i 个员工对第 j 个区域的喜好度。
答案输出：
对于每个测试数据，输出可以达到的最大的喜好程度。
输入样例：
3 3
1 1 1
100 50 25
100 50 25
100 50 25
输出样例：
175
数据解释：此数据只存在一种安排方法，三个员工分别安置在三个区域。最终的喜好程度为 100+50+25=175

2006 年百度之星程序设计大赛初赛题目 6
2007-05-14 17:42
百度语言翻译机
时限 1s
百度的工程师们是非常注重效率的，在长期的开发与测试过程中，他们逐渐创造了一套他们独特的缩率语。他们在平时的交谈，会议，甚至在各中技术文档中都会大量运用。
为了让新员工可以更快地适应百度的文化，更好地阅读公司的技术文档，人力资源部决定开发一套专用的翻译系统，把相关文档中的缩率语和专有名词翻译成日常语言。
输入数据：
输入数据包含三部分
1. 第一行包含一个整数 N （ N<=10000 ），表示总共有多少个缩率语的词条。

2. 紧接着有 N 行的输入，每行包含两个字符串，以空格隔开。第一个字符串为缩率语（仅包含大写英文字符，长度不超过 10 ），第二个字符串为日常语言（不包含空格，长度不超过 255 ） .
3. 从第 N+2 开始到输入结束为包含缩略语的相关文档。（总长度不超过 1000000 个字符）
输出数据：
输出将缩率语转换成日常语言的文档。（将缩率语转换成日常语言，其他字符保留原样）
输入例子：
6
PS 门户搜索部
NLP 自然语言处理
PM 产品市场部
HR 人力资源部
PMD 产品推广部
MD 市场发展部
百度的部门包括 PS ， PM ， HR ， PMD ， MD 等等，其中 PS 还包括 NLP 小组。
输出例子：
百度的部门包括门户搜索部，产品市场部，人力资源部，产品推广部，市场发展部等等，其中门户搜索部还包括自然语言处理小组。
注意：
1 ． 输入数据中是中英文混合的，中文采用 GBK 编码。
2 ． 为保证答案的唯一性，缩率语的转换采用正向最大匹配（从左到右为正方向）的原则。
请注意输入例子中 PMD 的翻译。

2006 年百度之星程序设计大赛试题

复赛题目

另类杀人游戏

周末的晚上，百度的员工们总喜欢聚集在公司的会议室玩杀人游戏。从 1 警 1 匪到 n 警 n 匪，他们尝试了几乎所有流行的杀人游戏规则。终于有一天，连最热衷杀人游戏，“ 杀人 ”不眨眼的 Austin 也开始对无休止的辩论感到厌烦。于是，他决定改变他的一贯作风，他开始变成了一个“ 杀人不睁眼 ”的杀手。

如何做到杀人不睁眼呢？ Austin 早已构思好他的杀人计划：

1 ． N 个人（包括 Austin ）坐成一圈玩杀人游戏，按顺时针编号 1 ， 2 ， 3 ， 4 。。。。。

2 ． Austin 从 1 号开始顺时针开始数到第 m 号就杀掉第一个人。被杀掉的人要退出游戏。

3 ． 如果第 m 个人恰好是 Austin 自己，他就杀掉他顺时针方向的下一个人。

4 ． Austin 从被杀的人的下一个顺时针数 m 个人，把第 m 个杀掉。

5 ． 重复 2-4 ，直至杀掉所有人。

Austin 把这个杀人计谋告诉了法官小 k ，他便可以闭起眼睛杀人啦。作为一个正直善良的法官，小 k 当然不能让残忍的 Austin 得逞，于是，她偷偷把 Austin 的杀人计划告诉了作为警察的你，聪明的百度之星。现在，你的任务是活到最后，与 Austin 单挑。

输入：

第一个行包含一个整数 T ，表示有 T 组测试数据。

对于每组测试数据：

三个整数

N ， M ， T ， (3<=N<=10000,1<=M,T<=10000) 分别表示参与游戏的人数， Austin 每隔 M 个人会杀掉一人， Austin 初始位置的标号。

输出：

每个测数数据输出一个整数。

你需要选择的初始位置的序号，以确保最后剩下的两个人是你与 Austin 。

输入例子：

2

7 4 1

7 4 1

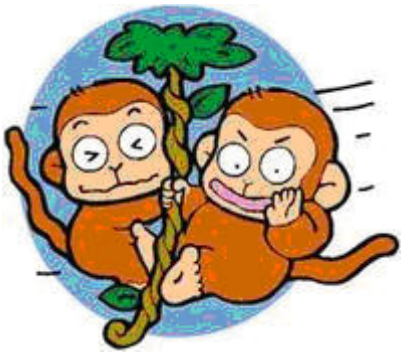
输出例子

5

5

例子说明：杀人顺序为 4 2 7 6 3 5 ， 所以 5 是你要选择的位置。

2006 年百度之星程序设计大赛复赛题目 **2**



空中飞猴

马戏团里新来了一只很特别的小猴子皮皮 —— 不仅长得漂亮，还很聪明。自从它来到马戏团之后，“ 空中飞猴 ” 成了马戏团里保留节目，慕名观看的人络绎不绝。“ 空中飞猴 ” 表演开始时，空中架着两根长长的钢丝。皮皮在其中一根上，它的目标是到达另一个根钢丝上。皮皮必须在爬行一定距离后纵身一跃，直接跳到另一根钢丝的某个位置。由于皮皮的速度非常快，它的运动轨迹可以近似的看成一条直线段。为了不让自己太危险，皮皮希望自己的跳跃距离尽量短，而为了不让观众等得太不耐烦，它在钢丝上的爬行距离不能超过 d 。在爬行距离不超过 d 的情况下，皮皮的跳跃距离最短是多少？

输入格式：

输入文件包含多组测试数据。每组测试数据包含 16 个实数 x1 ， y1 ， z1 ， x2 ， y2 ， z2 ， x3 ， y3 ， z3 ， x4 ， y4 ， z4 ， xp ， yp ， zp ， d ，表示两根钢丝分别为线段 (x1,y1,z1)-(x2,y2,z2) 和 (x3,y3,z3)-(x4,y4,z4) ，皮皮的坐标为 (xp, yp, zp) ，最大爬行距离为 d 。皮皮保证在第一条钢丝上，保证每条钢丝长度大于零。但两条钢丝有可能相交甚至重叠。

输出格式：

每组测试数据输出一行，仅包含一个非负实数，四舍五入保留三位小数，即最短跳跃距离。

输入样例：

0.0 0.0 0.0 4.0 4.0 0.0 4.0 0.0 1.0 0.0 4.0 1.0 2.0 2.0 0.0 10.0

输出样例：

1.000

说明：

共有 3 个测试数据集，每个测试数据集为一个输入文件，包含多组测试数据。每个测试数据集从易到难分别为 30 、 30 和 40 分，对每个测试数据集分别执行一次程序，每次必须在运行时限 3 秒内结束程序并输出正确的答案才能得分。

所有数据均从标准输入设备（ stdin/cin ）读入，并写出到标准输出设备 （ stdout/cout ）中。

三个测试数据集各有 10000 组测试数据。

2006 年百度之星程序设计大赛复赛题目 3

星球大战

公元 4999 年，人类科学高度发达，绝大部分人都已经移居至浩瀚的宇宙，在上千颗可居住星球上留下了人类的印记。然而，此时人类却分裂成了两个联盟：正义联盟和邪恶联盟。两个联盟之间仇恨难解，时有战争。

现在，正义联盟计划要破坏邪恶联盟的贸易网络，从而影响邪恶联盟的经济状况，为下一次战争作好准备。邪恶联盟由数百颗星球组成，贸易通过星球间的运输航道来完成。一条运输航道是双向的且仅连接两个星球，但两个星球之间可以有多条航道，也可能没有。两个星球之间只要有运输航道直接或间接的相连，它们就可以进行贸易。正义联盟计划破坏邪恶联盟中的一些运输航道，使得邪恶联盟的星球分成两部分，任一部分的星球都不能与另一部分的星球进行贸易。但是为了节省破坏行动所需的开支，正义联盟希望破坏尽量少的运输航道来达成目标。请问正义联盟最少需要破坏多少条运输航道呢？

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为两个整数 N 和 M（ $2 \le N \le 500$ ， $0 \le M \le N(N-1)/2$ ），N 为邪恶联盟中星球的数量。接下来 M 行，每行三个整数 A 、B 和 C（ $0 \le A$ ， $B<N$ ， $A \ne B$ ， $C>0$ ），表示星球 A 和星球 B 之间有 C 条运输航道。运输航道的总数量不超过 10^8 。

输出格式：

每组测试数据输出一行，包含一个整数，表示需要破坏的运输航道的数量。

如果输入的贸易网络本来就是不连通的，则输出 0 。

输入样例：

3 3

0 1 1

1 2 1

2 0 1

4 3

0 1 1

1 2 1

2 3 1

8 14

0 1 1

0 2 1

0 3 1

1 2 1

1 3 1

2 3 1

4 5 1

4 6 1

4 7 1

5 6 1

5 7 1

6 7 1

4 0 1

7 3 1

输出样例：

2

1

2

说明：

共有 5 个测试数据集，每个测试数据集为一个输入文件，包含多组测试数据。每个测试数据集从易到难分别为 5 、 10 、 15 、 30 和 40 分，对每个测试数据集分别执行一次程序，每次必须在运行时限 10 秒内结束程序并输出正确的答案才能得分。

所有数据均从标准输入设备（ stdin/cin ）读入，并写出到标准输出设备 （ stdout/cout ）中。

五个测试数据集中输入 N 分别不大于 20 、 50 、 100 、 200 和 500 ，各有 9 组测试数据。

2006 年百度之星程序设计大赛复赛题目 **4**

彩球游戏

X 博士是一个研究儿童智力开发方法的科学家，他为幼儿教育领域做出了许多贡献。最近， X 博士正在研究一种适合儿童的游戏，用以辅助发展儿童的观察力、注意力和思维能力。经过连日的构思， X 博士终于设计出了一种游戏：彩球游戏。

彩球游戏是一种单人参与的游戏，游戏首先给出一串由许多不同颜色的小球组成的小球序列，以及一个整数参数 M （ $M \geq 2$ ）。一段连续的具有相同颜色的小球序列称为连续同色序列。小孩，即游戏参与者，每次可以向任意一段连续同色序列插入一个同色小球，使该序列的长度加一。当一段连续同色序列在插入一个同色小球后其长度达到 M 时，该序列就会爆炸消失，然后原序列两边的其余小球会重新连成一串，如果两段相同颜色的连续同色序列在此时连接在一起，它们就会合并形成一段新的连续同色序列。如果新形成的连续同色序列长度达到 M ，这段序列也会爆炸消失，然后重复上述过程，直到没有新的长度达到 M 的连续同色序列出现为止。游戏的目标很简单，就是插入尽量少的小球，使得所有小球都爆炸消失掉。

通过长时间的游戏和不断提高游戏水平，这个游戏可以很好地开发儿童的观察力、注意力和思维能力。但是 X 博士仍然面临着一个困难的问题，他还需要设计出一个游戏演示 AI 程序，可以以最优的方式（即插入的小球数量最小）进行游戏，用于游戏教学，或者在游戏中对小孩给出提示。 X 博士并不擅长此类程序，因而他无法完成这个任务，你可以帮助他吗？

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为整数 M （ $2 \leq M \leq 20$ ），第二行为一条非空的字符串，由大写字母组成且长度不超过 200 ，表示初始的一串小球，不同的字母表示不同的小球颜色。初始时可能会存在一些长度达到 M 的连续同色序列，但这些序列不会马上爆炸消失。

输出格式：

每组测试数据输出一行，表示至少需要插入多少次小球才能使所有小球爆炸消失掉。

输入样例：

3

AAABAAA

3

ABBABBA

输出样例：

2

2

说明：

共有 5 个测试数据集，每个测试数据集为一个输入文件，包含多组测试数据。每个测试数据集从易到难分别为 5 、 10 、 15 、 30 和 40 分，对每个测试数据集分别执行一次程序，每次必须在运行时限 30 秒内结束程序并输出正确的答案才能得分。

所有数据均从标准输入设备（ stdin/cin ）读入，并写出到标准输出设备 （ stdout/cout ）中。

五个测试数据集中输入初始小球队列的长度分别不大于 10 、 20 、 50 、 100 和 200 ，各有不超过 5000 组测试数据。

2006 年百度之星程序设计大赛复赛题目 **5**

追捕

四个小孩正在花园里玩追捕游戏。一个小孩扮演逃亡者，其余三个小孩做追捕者。花园是一块由 N 行 M 列方格组成的草地，花园周围有木栏包围着，不能走出，花园里面还有一些障碍物不能够通过。游戏可以进行许多回合，每个回合分成两轮，第一轮追捕者可以进行追捕行动，第二轮逃亡者可以根据前一轮追捕者的行动开展逃亡旅程。在第一轮里，三个追捕者必须在三人中选择一个人向某个相邻的方格走一步，只有在三个人都没有可以走的相邻方格时，他们才允许选择停留在原地。在第二轮里，逃亡者也必须选择某个相邻的方格走一步，如果逃亡者没有任何可走的方格，那么逃亡者就被捕了。四个小孩都不允许走到有障碍物或其他人的方格上，也不能走出花园，因而，四个小孩总是会位于不同的方格上面。

这些小孩都是非常聪明的，三个追捕者也是团结一致的。追捕者如果有可以捉到逃亡者的方法，那么他们就一定不会错过。逃亡者如果有不被捕获的方法，那么他也不会犯错。除此之外，追捕者会希望尽快地捉到逃亡者，而逃亡者即使在会被捕获的情况下也会尽可能地拖延时间。给定花园的障碍物的分布图和四个小孩的初始位置，你知道追捕者有方法捉到逃亡者吗？如果有，他们要经过多少轮后才能捉到逃亡者呢？

输入格式：

输入文件包含多组测试数据。每组测试数据第一行为两个整数 N 和 M （ $1 \le N \le 10$ ， $1 \le M \le 10$ ），为花园方格阵列的行数和列数。接下来 N 行，每行 M 个字符，可以为“.”、“#”、“O”和“X”，分别表示空地、障碍物、追捕者和逃亡者。追捕者总是会有三个，而且四个小孩一开始也都会在空地上面。

输出格式：

每组测试数据输出一行，若追捕者能够捉到逃亡者，则输出他们要经过多少轮后才能成功。轮数的计算包括追捕者和逃亡者进行行动的两轮，逃亡者被捕获的那一轮不算，因而结果总是一个奇数。具体输出格式请参考输出样例。

输入样例：

2 2

OO

OX

3 3

OOO

##X

...

3 3

OO#

###

.OX

3 4

OO##

####

..OX

4 4

OOO.

....

....

...X

5 5

O...O

.....

..#..

.....

O...X

5 5

O...O

.....

...#.

.....

O...X

6 6

.....

.O..O.

..##..

..##..

.O..X.

.....

6 6

#.....

.O..O.

..##..

..##..

.O..X.

.....

10 10

....

....

..O....O..

....

....

....

....

..O....X..

....

....

10 10

....

.#.#.#.#.#

..O.....O.

.#.#.#.#.#

....

.#.#.#.#.#

....

.#.#.#.#.#

..O.....X.

.#.#.#.#.#

输出样例：

The escapee will be captured after 1 steps

The escapee will be captured after 7 steps

The escapee will be captured after 5 steps

The escapee will never be captured

The escapee will be captured after 21 steps

The escapee will never be captured

The escapee will be captured after 41 steps

The escapee will never be captured

The escapee will be captured after 39 steps

The escapee will never be captured

The escapee will be captured after 51 steps

说明：

共有 5 个测试数据集，每个测试数据集为一个输入文件，包含多组测试数据。每个测试数据集从易到难分别为 5 、 10 、 15 、 30 和 40 分，对每个测试数据集分别执行一次程序，每次必须在运行时限 60 秒内结束程序并输出正确的答案才能得分。

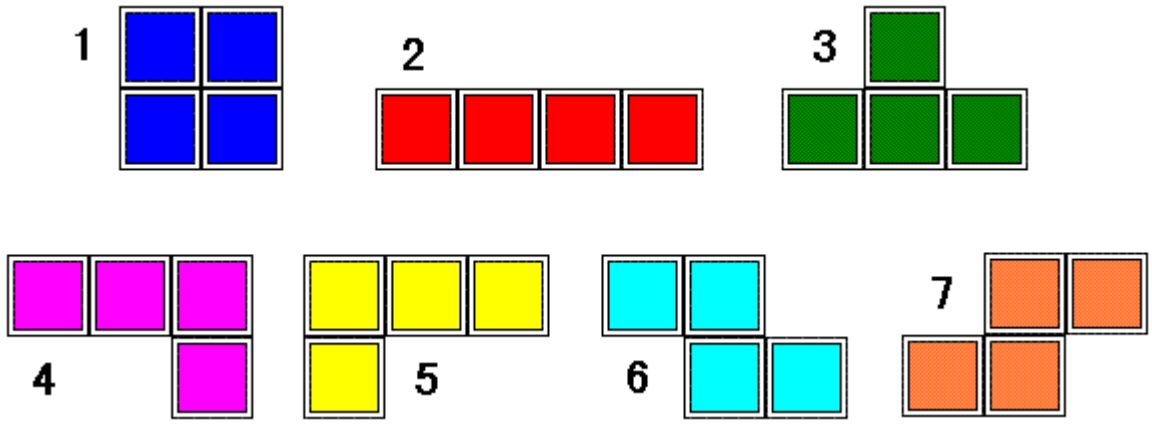
所有数据均从标准输入设备（ stdin/cin ）读入，并写出到标准输出设备 （ stdout/cout ）中。

五个测试数据集中输入 N 、 M 分别不大于 6 、 7 、 8 、 9 和 10 。

2006 年百度之星程序设计大赛试题总决赛题目

俄罗斯方块

俄罗斯游戏中共有七种方块，每种方块都由四个方格组成，如下图所示，七种方块分别编号为 1~7 。



游戏中,每次落下一个方块,落到一个宽度为 10 格的槽中。方块的下部一旦碰到槽的底部,或槽中已有的方块,就不能再移动。方块落下不动后,如果有某些行因落下的方块而填满,这些行将被消去。方块下落前,你可以控制方块的左右移动和旋转,以将其放在合适的位置。你对方块的所有移动和旋转操作在下落前(槽外)就计算完毕,然后直接下落到底,下落过程中不能再做操作。如果方块刚刚落下后顶部高度大于 17 行,游戏结束 -- 即使此时有些行可以消除。

交互方式

你的程序应当包含 tetris_lib ,并连接相应的库文件。库中的两个重要函数是:

void int

int

你的程序应该首先调用 StartGame ,其中 t1 和 t2 表示前两个方块的编号(t2 对应于传统游戏中的 ″ 下一个方块 ″)。接下来,你的程序每次可以使用 Ste 函数下落一个方块,返回消去的行数。r 表示旋转方式(r=0, 1, 2, 3 分别表示顺时针旋转 0 度、90 度、180 度、 270 度), l 表示方块在旋转后的最左边一格的列编号(从左到右依次为 1, 2, … , 10),而 next 表示方块落下后新的下一个方块编号(0 代表没有下一个方块,下一次 Step 调用后库将自动终止你的程序)。你的程序不应自行终止。

关于自测的提示

调用 StartGame 函数时,库将从标准输入中读入若干行,每行包括一个整数,表示方块的编号。你可以利用这一点对你的程序进行测试。程序运行结束后,测试库将把结束原因和得分显示在标准输出中。

库中还有两个函数可以用于自测：

void

void

如果需要测试库记录程序的行为，请在调用 StartGame 之前调用 SetLog 函数。

评分规则

在与测试库交互的过程中，出现以下条件之一，则库将终止程序：

1 、调用参数非法

2 、方块刚刚落下后，其顶部高度大于 17 行

3 、所有方块均已落下

4 、本数据的运行总时间超过 1 秒钟

程序终止后，假设一次消去 1 行、 2 行、 3 行、 4 行的次数分别为 a, b, c, d ，则该数据原始得分为 2b+ 6c +10d 。换句话说，消去单独的 1 行不得分。

对于每个数据，得分排名前八的程序分别得到 10, 7, 6, 5, 4, 3, 2, 1 分。如果原始得分相同，则消去行数多的排名在前；如果原始得分和消去行数都相同，则下落方块总数多的排名在前。如果三者都相同，则得分相同。消去行数为 0 的程序不得分，即使它排在前八。

最终成绩按照 50 个数据的总分从大到小排序，如果总分相同则按所有数据的原始得分之和排序；如果仍有相同，则按所有数据的消去的总行数排序；如果仍相同，则名次相同。

2007 年百度之星程序设计大赛试题

复赛题目

好心的出租车司机
题目描述

北京的一位出租车司机向你抱怨：城市发展太快，公路越来越多，他已经疲于计算行驶路线，于是求助你开发一个自动导航的工具。

出租车只能在公路上行驶。所有的公路都是笔直、双向的，相交的公路视为连通（可以在交叉点处从一条公路开到另一公路上）。由于道路施工，整个城市的公路系统可能并不完全通畅。如果乘客的目的地不在公路边，则乘客下车后要步行前往，步行路线不受公路限制。这位好心的司机还特别提出，乘客步行距离越短越好；其次，出租车行驶里程越短越好。

方便起见，用笛卡尔坐标系来描述城市地图，所有坐标都在第一象限［0, 1000］的范围内。公路宽度忽略不计。

输入格式

第一行是一个数字 k ，代表公路条数。以下 k 行每行用 4 个实数描述一条公路（用空格隔开），前两个表示公路起点，后两个表示公路终点。下一行包含 4 个实数（用空格隔开），前两个表示乘客上车点，后两个表示乘客目的地坐标。不相交公路间的最短距离至少为 10^{-4} 。

输出格式

仅一行，为出租车行驶的里程数，保留一位小数。

输出格式

2

2.0 2.0 10.0 10.0

10.0 2.0 2.0 10.0

3.0 3.0 9.0 4.0

输出样例

7.8

评分方法

本题有 20 组数据，满足 k<=100, 公路的交点数不超过 10000 。

Robots.txt 协议

题目描述

搜索引擎是靠 Web Robot （又称 Spider ）来收集互联网上浩如烟海的网页的。 Spider 就像一个旅行家一般，不知疲倦地奔波于万维网的空间，将遇到的页面收集下来供搜索引擎索引。对于一个网站的管理员来说，如果希望搜索引擎只收录自己指定的内容，或者指定某些不希望搜索引擎访问的内容，该如何去做呢？他需要的就是 Robots Exclusion Protocol 协议，这里简单的称它做 Robots.txt 协议。

Robots.txt 是一个放置在网站根目录下的纯文本文件。举例来说，当 Spider 访问一个网站（比如 http://www.example.com ）时，首先会检查该网站中是否存在 http://www.example.com/robots.txt 这个文件，如果 Spider 找到这个文件，它就会根据这个文件的内容，来确定它访问权限的范围。

www.robotstxt.org 是 robots.txt 协议的 Home Page ，在这个站点上你可以找到很多 robots.txt 协议相关的资料。 Robots.txt 协议在 http://www.robotstxt.org/wc/norobots.html 上有比较详尽的描述。

你的任务就是编写 Spider 中的一个逻辑单元，这个单元的作用就是来判断一个网站的一些 URL 是否被禁止抓取。对方网站的站点在这里假设是 www.example.com ，这个 Spider 的 User-agent 当然是 Baiduspider 。注意，这个逻辑单元除了本身的判断逻辑要求与 robots.txt 协议一致外，还要注意容错的问题。互联网上纷繁芜杂，会出现很多意想不到的错误。如何能够对一个对错参半的 robots.txt 进行解析，把其中正确的挑拣出来、把错误的部分忽略掉，也是一个不小的挑战哦。都会遇到什么错误？在开始爬行互联网之前，谁都不知道。

输入格式

第一行是一个正整数 m ，表示 robots.txt 文件的行数，后面跟 m 行，是 robots.txt 的全文。下一行包含一个正整数 n ， 表示 URL 的行数，后面跟 n 行 URL ，这个就是你要判断的 URL 的列表。

输出格式

每条 URL 输出一行，每行两列，第一列是一个数字，如果这条 URL 被禁止，则输出 0 ，否则输出 1 。第二列是这条 URL 本身。

输入样例

2

User-agent: *

Disallow: /tmp/

2

http://www.example.com/index.html

http://www.example.com/tmp/somepage.html

输出样例

1 http://www.example.com/index.html

0 http://www.example.com/tmp/somepage.html

评分方法

本题包含 20 组数据，均满足 0<=n,m<=100 。

简单印象

题目描述

简单、可依赖是百度的性格，百度之星们又有怎样的性格呢？

有 n 名选手入围了百度之星程序设计大赛的复赛阶段。为了让选手相互之间有更多的了解和更好的交流，组委会工作人员邀请每位选手选择一些不同的词语来介绍自己的性格。每名选手需要为自己选定的每一个词语指定一个绝对值不超过 100 的整数权值 q ，表示这个词语在多大程度上描述了自己的性格。例如“美丽 90 ”表示该选手认为自己非常美丽，而“冷淡 -60 ” 表示比较热情，而“外向 -5 ” 表示稍微有一点点偏内向。你不需要考虑词语本身的语义，比如“外向 -5 ” 不代表“内向 5 ”。

组委会发现不少选手的性格都有相似之处，所以希望找到一组词语尽量准确的描述所有的选手。对于选出的词语集合 S ，定义选手 i 被描述的精确程度 P(i) 为 S 中所有词语在选手 i 眼中的权值之和（选手 i 没有提到的词权值视为 0 ），组委会希望让最小的 P(i) 尽量大，因为这样才能让选出的词语能够描述所有选手，而不仅仅是部分。

另外，所选词语的个数也是有讲究的。词语太少会略显单薄，而词语太多又不方便宣传，所以组委会设定了词语个数的最小值 min 和最大值 max 。

输入数据

输入数据第一行为三个整数 n, min 和 max ，表示选手的个数、词语的最小值和最大值，接下来的 n 行，每行包括若干“词语 - 权值”对。词语保证由不超过 10 个汉字组成（用 GBK 编码，不含非汉字），权值为绝对值不超过 100 的整数。

输出数据

仅一行，输出各个词语，用空格隔开。次数总数必须不小于 min 且不大于 max ，每个词语都必须至少被一个选手提到过，否则视为非法输出。

输入样例

3 2 4

美丽 90 冷淡 -60 外向 -5 乐观 20

美丽 10 冷淡 20

外向 20 乐观 -5

输出样例

美丽 冷淡 外向

样例解释

如果把选手按照输入中出现的顺序编号为 1~3 ，则 P(1)=90-60-5=25 ， P(2)=10+20=30 ， P(3)=20 ， 所有 P 中的最小值为 20 。

评分方法

本题包含 30 个测试点，每个测试点 10 分，共 300 分。

测试点 1~10 满足 n<=100, 1<=min<=max<=5, 涉及到的词语不超过 500 个。

测试点 11~20 满足 n<=200, 10<=min<=max<=15, 涉及到的词语不超过 2000 个。

测试点 21~30 满足 n<=400, 25<=min<=max<=30, 涉及到的词语不超过 10000 个。

每个测试点独立评分。对于每个测试点，非法输出的得分为 0 ， 合法输出的得分大于 0 。设合法输出的程序数为 M ，比程序 i 的方案严格更优的程序数为 Y(i) ，则该测试点程序 i 的分值为 10(1-Y(i)/M) 。换句话说，输出该测试点最优解的程序将获得 10 分，而最差解惟一的情况，输出最差解（但合法）的选手将得到 10/M 分。注意：每个测试点的得分不必为整数。

紧急修复

背景

2050 年的一天，某市 k 家公司的计算机系统突然同时瘫痪。市政府很快找到了问题的所在，并开始研发一套自动修复整个城市的系统。自动修复系统启用后整个城市的计算机系统将恢复正常，但由于研发时间较长，在此之前各公司仍将蒙受不小的损失。为此，市政府决定派出 n 支维修能力相同的维修队到各公司进行手工修复，力图在自动修复系统启用前整个城市的总损失达到最小。

参数

城市被划分成 R*C 的网格，各行从上到下编号为 1~R ，各列从左到右编号为 1~C 。每个格子为空地、障碍物和建筑物之一（公司总是位于某个建筑物格子中）。维修队每次只能往上（行编号减 1 ）、下（行编号加 1 ）、左（列编号减 1 ）、右（列编号加 1 ）四个方向移动，第 i 个维修队每个小时可以移动 s(i) 格。维修队在任何时候都不能位于障碍物中，但建筑物可以从它上下左右的相邻空地进入，也可以从这些格子出去。注意：不能直接从一个建筑物格移动到另一个建筑物格，而必须先移动到相邻的空地，在空地上移动，然后再进入另一个建筑物。每个格子的实际尺寸很大，因此可以有多个维修队同时在同一个格子中。

第 i 家公司的位置为 (r(i), c(i)) ，瘫痪程度为 B(i) ，每小时的经济损失为 P(i) 元。瘫痪程度的单位是“队•小时”，即一支维修队每小时可以把一个公司的瘫痪程度降低 1 ，而如果 m 支维修队同时为一个公司修复，每小时可以把该公司的瘫痪程度降低 m 。

注意，在完全修复之前，每个小时的经济损失不变。

修复计划

政府的修复计划应包含每个小时各维修队所执行的命令。这些命令包括：

1. REST

该命令不进行任何操作。

2. MOVE〈seq〉

按照 seq 进行移动。其中 seq 为一个长度不超过 s 的非空字符串（如果不需要移动，请使用 REST 命令）。如果 seq 的长度超过 s ，则从第 s+1 个字符开始的剩余部分将被删除；如果该命令不能完全成功的执行，则从第一个非法移动开始的所有后续移动将被忽略。

3. REPAIR

对当前公司进行维修。如果当前公司已经修复或者当前位置不是公司，该命令将被忽略。该命令后面加的所有参数将被忽略。

需要注意的是，每个小时只能执行一条指令，例如不能在 MOVE 之后立刻 REPAIR ，必须等待下一个小时。

输入格式

输入的第一行为三个正整数 R, C, T 即网格的行数、列数和自动修复系统的启用时间。以下 R 行每行 C 个字符，表示该城市的地图。点表示空地， # 表示障碍物， O 表示建筑物。

下一行包含一个正整数 k ，表示公司的数目。以下 k 行每行四个整数 r, c, B, P ，其中 (r,c) 表示该公司坐标 (1<=r<=R, 1<=c<=C) ， B 为该公司的初始瘫痪程度， P 为每小时的经济损失。 (r,c) 处保证为一个建筑物格。 B 和 P 保证大于 0 。

下一行包含两个正整数 n, s ，表示维修队的个数和每小时最多移动的格子数。以下 n 行每行包含两个整数 r, c ，表示该维修队的初始位置。维修队按照输入文件中的顺序编号为 1~n 。

输出格式

输出每 n 行为一组，描述一个小时各维修队的发出的命令。共 T 组，一共 nT 行。所有格式非法的指令将被忽略(等效于 REST)。尽管如此，如果程序输出的命令中没有 REPAIR ，或者所有的 REPAIR 都没有成功执行，则输出将视为非法。

输入样例

```
4 7 5
...#O##
#.....#
O...##O
#......
2
1 5 3 5
3 7 4 6
3
4 7 5
1 1 5
3 1 5
```

输出样例

```
MOVE U
MOVE RRRD
MOVE RDRURD
REPAIR
```

MOVE DRRU

MOVE DRRRU

REPAIR

REPAIR

REPAIR

REPAIR

REPAIR

MOVE D

REST

REST

REPAIR

模拟器

每小时的模拟过程如下：

第一步： 对于每个尚未修复的公司 i ，将 P(i) 累加进总损失。

第二步： 按照序列从小到大的顺序依次执行各维修队的指令。

为了更清晰的说明规则并帮助选手设计和测试程序，命题组开发了一个简单的模拟器。该模拟器根据输入数据和程序输出进行模拟，给出详细模拟过程，以及最后的结果。最终的测试将严格按照该脚本的输出进行评判。

模拟器用 python 编写，没有安装 python 的选手可以在 http://www.python.org 下载最新版本。

评分方法

本题包含 30 个测试点，每个测试点 10 分，共 300 分。

测试点 1~10 满足 R, C<=10, n<=5, k<=10, B<=15, T<=30

测试点 11~20 满足 R, C<=30, n<=10, k<=20, B<=50, T<=500

测试点 21~30 满足 R, C<=100, n,<=100, k<=500, B<=1000, T<=10000

每个测试点独立评分。对于每个测试点，非法输出的得分为 0 ，合法输出的得分大于 0 。设合法输出的程序数为 M ，比程序 i 的方案严格更优的程序数为 Y(i) ，则该测试点程序 i 的分值为 10(1-Y(i)/M) 。换句话说，输出该测试点最优解的程序将获得 10 分，而最差解惟一的情况，输出最差解（但合法）的选手将得到 10/M 分。注意：每个测试点的得分不必为整数。

2007 年百度之星程序设计大赛总决赛题目

比赛目标：
完成基于 CounterStrike1.5 的 podbot AI，在 fy_iceworld_plus, de_dust2, de_inferno 三张地图上与对手 AI 展开较量。Podbot 的基本代码框架由平台给出，请选手自行参考研究。
有待完成的代码位于 bot_T.cpp bot_T.h bot_CT.cpp bot_CT.h 中，各个函数功能与接口请选手参考代码中的注释说明。
比赛方式：
1． 提交代码：请将 bot_T.cpp bot_T.cpp bot_CT.cpp bot_CT.h 压缩成一个 rar 压缩文件,通过比赛网站提交：
第一二组使用 http://192.168.0.3/
第三四组使用 http://192.168.0.4/
第五六组使用 http://192.168.0.5/
2． 赛制：
第一阶段：从比赛开始至之后 4 小时整。提交第一阶段 bot 代码用于 iceworld 测试，测试结果将决定第二阶段分组。
第二阶段：使用 dust2 和 inferno 两张地图，赛程是由小组循环决出前八。前八由淘汰赛决定胜负，判定最终名次。
操作说明
编译
进入目录 C:/astar2007/MSYS
双击运行 msys.bat
在 msys 界面中输入 cd /c/astar2007/PODBOT
执行命令 make ，如果 make 成功，则在.obj.win32 子目录中有 podbot_mm.dll 生成
调试
1． 在代码中使用 UTIL_ServerPrint 可以向 console 打印信息在游戏中使用

2． 可以记录 log 文件以供分析检查之用
运行
1． 将 podbot_mm.dll 拷贝到 C:\Program Files\反恐精英中文站 CS1.5 中文硬盘版\CS1.5 中文硬盘版\cstrike\addons\podbot 覆盖原有文件即安装成功。
2. 双击桌面上的 cstrike 快捷方式，启动游戏， 单击进入游戏，建立一个 11 人局域网游戏。进入游戏后按 6 进入观察模式
3．控制 bot
(1) 按 =7 杀掉所有自动 bot
(2) 按 =5 添加 bot ， 选择 godlike 且 aggressive 的 bot
(3) 观察游戏结果 按 ~可以观察 Server 调试信息输出
规则说明：
0． 裁判组拥有最终裁定权
1． 胜负判定原则
（1） 地图为炸弹图时，如 de_dust2 和 de_inferno 时， 以炸弹是否成功爆炸为胜负条件。当地图为 fy_iceworld_plus 时以杀伤数为胜负条件。
（2） 当由于赛程问题导致平局出现时，由裁判组判定
2． 违例判定原则
（1） AI 程序不正常导致程序无法运行时，经裁判组认定后判负
（2） 以各种形式 hack 对手数据，经裁判组认定后判负

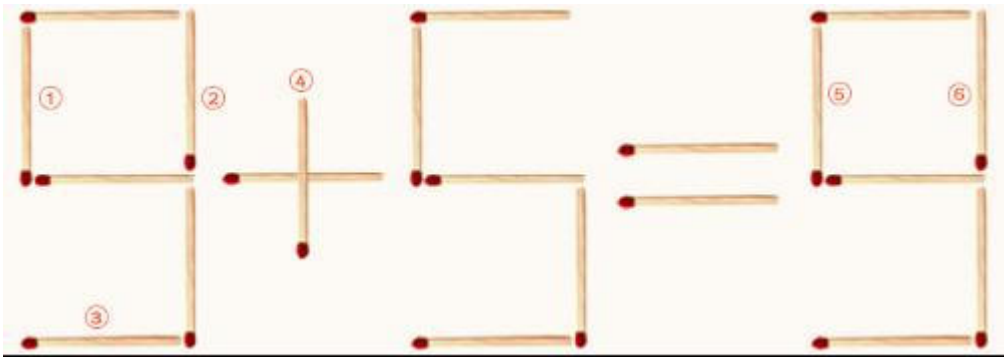
百度之星 2009 程序设计大赛 初赛第一场试题

2009 年 5 月 30 日 19：00-22:30（由于第二题出错，比赛时间延长半小时），2008 百度之星大赛在线资格赛（初赛）展开。百度爱好者（Baiduer.com.cn）在第一时间给大家带了初赛题目。第一场初赛共四题，分别是火柴游戏(250 分)、电子商务平台商品推荐问题 （300 分）、争车位（300 分）和葫芦娃 （350 分），总计 1200 分。

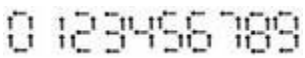
1． 火柴游戏(250 分)

题目描述

在百度，同事们之间喜欢交流游戏。其中，火柴游戏是一个比较经典的例子。游戏的规则很简单：恰好移动一根火柴，使等式成立。如下面的等式可以变成 3+6=9（还有其他解）：移动哪一根火柴能使等式成立?



下面是所有火柴数字的样子



请你写一个程序，找出所有的规范解。所谓规范是指：

＊ 只能改变数字，不能改变符号；

＊ 数字和符号的组成方式必须严格的和图示的一样（减号由一根火柴组成）；

＊ 新等式必须形如 a+b=c 或 a-b=c，其中 a、b、c 都是不含前导 0 的非负整数。

当然，最重要的是：新的等式必须在数学上成立。

输入格式

输入仅一行，为一个格式为 a+b=c 或 a-b=c 的表达式，其中 a、b、c 均为不含前导 0 的非负整数。表达式长度不超过 100，且不含空白字符。因此，加号/减号紧跟在 a 的后面、 b 紧跟在加号/减号的后面、等号紧跟在 b 的后面、c 紧跟在等号的后面。

输出格式

输出所有规范解，按字典序输出（请注意：输出顺序不对将不得分）。无解时，仅输出一行-1。

样例输入 1

9+5=9

样例输出 1

3+5=8

3+6=9

样例输入 2

1+1=2

样例输出 2

-1

测试数据

共 10 个测试点，基本参数如下表：

测试点编号	表达式的长度
1-2	1-10
3-4	10-25
5-6	26-50
7-10	51-100

裁判问答：

Q:第一题的表达式长度不是只有 5 吗?　A:可以多位整数

Q：把某根移出来再移到原来的位置上算不算移动? A：不算

2. 电子商务平台商品推荐问题 （300 分）

题目描述

百度网络交易平台（“百度有啊”） 是建立在百度旗下独有的搜索技术、强大社区资源基础上的中文互联网领域最具规模的网上个人 C2C 交易平台。伴随着“百度有啊”的成长，“有啊”的顾客也蜂 拥而至；面对如此大量的用户，如何把平台上数以千万计的商品按一定的规则推荐给他们以促成交易，是“百度有啊”面临的重要问题。

在本题中，假设有 M 个用户和 N 种产品，每个用户的浏览历史可以用一个 N 维特征向量 X 描述：Xi=1 当且仅当该用户曾经浏览过商品 i。如果用户 A 和用户 B 曾浏览过(部分)相同的商品，我们说用户 A 和用户 B 相似；如果用户 A 和用户 B 相似， 或者用户 A 和一个“与用户 B 相似”的用户相似，则需要把用户 A 和用户 B 划分到同一个用户群。该用户群中所有用户的特征向量的“按位或” 便是整个用户群的特征向量——它表示至少被其中一个用户浏览过的商品集合。

每当一个新用户到来时，可以计算出它和所有用户群之间的相似度。假定他的特征向量为 A，用户群的特征向量为 B，则：

$$similarity(\vec{A},\vec{B}) = \cos(\vec{A},\vec{B}) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

其中计算特征向量模长时使用的是二范数，即所有维度数值的平方和的算术平方根。

接下来，我们应找出该用户最接近的用户群，并从该用户群购买过的商品中选三个商品进行推荐。一般来说，一个商品被购买的次数越多，就越应该被优先推 荐（若购买次数相同，优先推荐 id 值小的商品），但为了避免马太效应，我们需要做一个特殊处理：不推荐最畅销的商品（如果有多个商品的购买次数都是最多的，则它们都不应该被推荐，为简单计，如果所有商品的购买次数都一样的话就都不推荐了）。另外，推荐给用户的商品不能是他已经购买过的商品。

如果从最接近的用户群中无法推荐出三个商品，应从次接近的用户群购买的商品中以相同的规则补充，以此类推，直到选出三个推荐商品，或者无法选出更多商品。

测试数据保证任何一个用户不会跟两个不同用户群的相似度相同，因此商品推荐的结果总是惟一的。

注意：如果用户浏览记录跟某用户群的相似度为 0，则在任何情况下都不从该用户群购买的商品中推荐。

输入格式

第 1 行：M、N，分别是平台用户数和商品总数。0 < M,N <= 100 000

第 2 行：K，表示接下来有 K 行记录。0 < K <= 100 000

第 3~K+2 行，每一行是一次用户的浏览-购买记录，记录格式为：

uid i1,i2, …,ip b1,b2,…,bq

其中 uid 是不超过 M-1 的非负整数，代表用户 id。i 为本次浏览的商品 id 集合（无重复元素，元素顺序无意义），而 b 为该用户在此次浏览后购买的商品集合（无重复元素，元素顺序无意义）。i 中的每个元素均为不超过 N-1 的整数，b 是 i 的子集。q <= p <= 50。注意：同一个用户 ID 可以对应多条记录

第 K+3 行：Q，表示接下来有 Q 次查询。0 < Q <= 125

第 K+4~K+Q+3 行：每一行是一次用户的浏览记录，格式为

uid i1,i2, …,ip

含义类似于浏览-购买记录。注意：用户群划分方式完全取决于第 3 行开始的 K 条记录。这里的查询不会导致用户群划分方式的变化（在实际的系统中，用户群数据也是定期更新，而非实时修改）。

输出格式

对每个查询输出一行,为推荐的最多三个商品的 id(为不超过 N-1 的非负整数)，按推荐度降序排列。如果没有任何可推荐的商品，输出 NONE；如果有商品可推荐，但不足三个，应全部输出。

样例输入

9 12

8

0 0,1,2 1

1 3,4,5 3,4

2 1,5 1

3 6,7 6,7

4 8,9 8

5 8,10 8,10

0 11 11

8 6 6

3

6 0,1,2

1 0,1,2,6

3 8,9

样例输出

3 4 11

11 7

10

PS：原样例有错，于 22：45 分修正样例

样例解释

首先对购买历史进行预处理形成用户群，然后对每个用户群购买的商品按购买次数进行排序，结果如下：

用户群 ID			
用户 ID	浏览过的商品 ID 集合	购买的商品 ID	
0		0,1,2,3,4,5,11	1 次),3,4,11
1	3,8	6,7	6(2 次),7
2	4,5	8,9,10	8(2 次),10

接下来处理查询。

输入：0,1,2　　输出：3,4,11

此浏览记录与用户群 0 最接近。根据规则推荐 3,4,11 （1 是最畅销商品，不推荐；对于被购买数量相同的商品 3 和 4，优先推荐 id 值小的商品）

输入：0,1,2,6　　输出：7,11

此浏览记录与用户群 1 最接近，根据规则推荐 7（6 是最畅销商品，不推荐）；由于不足三个商品，从次接近的用户群 0 中推荐，推荐 11（1 是最畅销商品，3、4 是购买过的商品）；仍然不足三个商品，但是已无更多商品可以推荐。

输入：8,9　　输出：10

此浏览记录与用户群 2 最接近，根据规则推荐 10；不足三个商品，但是已无更多商品可推荐。

测试数据

共 20 个测试点，基本参数如下表：

测试点编号	M	N	K	Q
1				
2				
3				
4				
5				
6	<=1000	<=1000	<=1000	<=100
7	<=1000	<=1000	<=1000	<=100
8	<=1000	<=1000	<=1000	<=100
9	<=1000	<=1000	<=1000	<=100
10				
11				
12				
13				
14				
15				
1				
1				
1				
1				
20				

3　争车位（300 分）

题目描述

争车位是目前 SNS 网站上比较热门的游戏之一。假如小明有 N 个好友和 M 辆车。每个好友都有 K 个车位。这些车位可能停放了小明的车，也可能停放了其它人的车，还能没有停任何车辆（空车位）。在当前状态中，小明的 M 辆车全停放在他的好友的车位上。

任务一：考虑如下的移车规则：

1. 必须按一定顺序依次移动自己的车，而不能移动别人的车。

2. 每辆车只可以移动一次。

3. 车只能从原来的位置移到空车位处，不能移动到一个已停放了车的车位（无论这辆车是谁的）。移动后，原来的位置就成了空车位。

4. 只能跨好友移车。也就是说，一辆车不能从某好友的车位移到这个好友的另一车位。

请帮助小明把他的所有 M 辆车都移动一遍。

任务二：考虑如下的移车规则：

1. 每个车位都对应一个金额。当小明把一辆车最终停在某个车位时，将会得到该车位对应的金额。

2. 不一定需要把所有车都移一遍，但只有移动前后处于不同车位的车才能得到对应的金额。

3. 可以把自己的车开到附近的空地上（那里足以停下他所有的车）作为临时中转。因此车不必依次移动，每辆车也可以多次移动。但请注意：所有移动结束之后，每个车位最多只能停一辆车。

4. 和任务一相同的是：仍不许动其他人的车，且每辆车在移动之前和所有移动结束后所在的车位仍必须属于不同好友。特别地，不许最终把一辆车停到用于中转的那个空地上。

请帮助小明获得最大的总金额。

输入格式

第 1 行包含一个整数 T，即任务编号（T=1 或 2）。

第 2 行包含两个整数 N、K，分别表示小明的好友数和每个好友的车位数。以下 N 行每行有一个包含 K 个字符的字符串，描述每个好友当前车位的状态。其中 ‘#’ 表示该车位停放的是小明的车；‘*’ 表示该车位停放的是其它人的车； ‘.’ 表示该车位是空车位。

如果 T=1，输入到此结束；否则接下来的 N 行每行包含 K 个不超过 100 的整数，分别表示每个车位对应的金额。

输入据至少有一辆小明的车。

输出格式

如果 T=1，输出任意可行的移车方案。假如有 M 个步骤，则第一行输出 M，紧接着 M 行表示 M 步的具体内容。格式为：(x1,y1)->(x2,y2)，表示把第 x1 个好友第 y1 个车位上的车移到第 x2 个好友的第 y2 个车位。如果无解，输出-1。

输入 T=2，输出一个数字，表示最多的金额总数。

样例输入 1

1

3 4

##*.

.#**

****

样例输出 1

3

(0,0)->(1,0)

(1,1)->(0,3)

(0,1)->(1,1)

样例输入 2

1

3 4

##*#

.#**

****

样例输出 2

-1

样例输入 3

2

2 2

#*

#*

2　1

1　2

样例输出 3

3

测试数据

共 20 个测试点，基本参数如下表：

测试点编号	T	N	K
1	1	3	6
2	1	10	10
3	1	10	10
4	1	3	4
5	1	100	100
6	1	10	100
7	1	10	100
8	1	10000	100
9	1	100000	100
10	1	100000	100
11	2	2	4
12	2	10	4
13	2	5	5
14	2	3	4
15	2	20	4
16	2	50	10
17	2	50	10
18	2	10	10
19	2	50	10
20	2	50	10

4. 葫芦娃 （350 分）

题目描述

蝎子精和蛇精为祸人间，葫芦七兄弟准备与之决一死战。不幸的是，七弟不慎被两只妖精抓住，困在了蛇蝎山的囚笼里，其余六兄弟必须尽快去营救。二娃使 用千里眼，查看到七弟被囚的位置，但蛇蝎山地势复杂，机关遍布，如何才能又快又安全的把七弟救出来呢？于是，六兄弟请您来帮忙了。

在二娃的帮助下，大家先绘制了一张蛇蝎山的地图，并把能安全停留的地方以点标记。你很快就发现，这些安全点组成了一个六邻接图——每个点都与左上、左、左下、右下、右、右上六个点等距。于是，你以其中两条坐标轴：“左——右”和“左上——右下”，给各点设置坐标（见图 1）。只要把囚笼的六个邻接点都占了，然后六兄弟一起施法，就能把七弟营救出来。

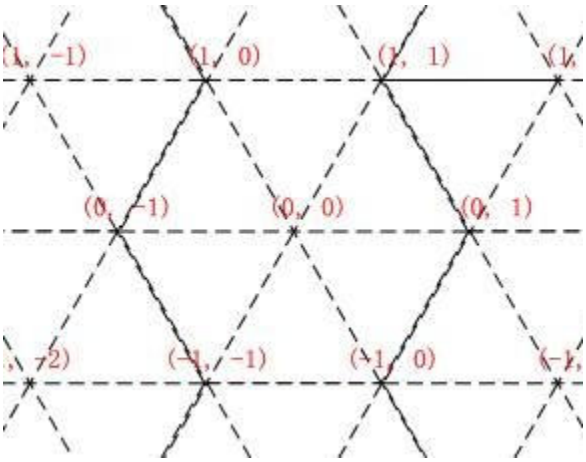


图 1. 六邻接图及坐标

虽然已经有了地图，但怎样走才能最快的把七弟救出来呢？葫芦兄弟告诉你，他们有两种移动方式：

1、跑步。可在一单位时间移动一单位距离，即从一个点移动到某个邻接点（见图 2）。

图 2. 跑步移动示意图

2、翻跟头。可在一单位时间沿着一个坐标轴方向移动多个单位距离，但其飞过的每个点上都必须有葫芦兄弟站在那里施法。例如，在点 (0,0)和点(1,1)都有葫芦娃，那么位于点(2,2)的葫芦娃便可在兄弟的帮助下，沿着“左下——右上”坐标轴直接翻跟头到点(-1,-1)（见图 3）。

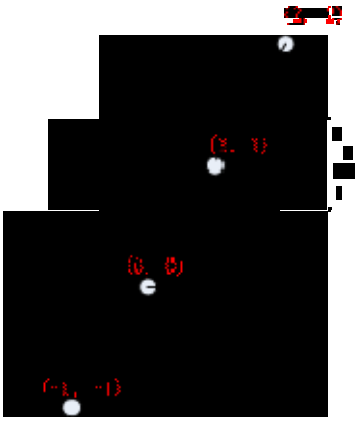


图 3. 翻跟斗移动示意图

另外，为了不引起妖精的注意，每一单位时间最多只有一个葫芦娃能移动，且每个点上只能站一个葫芦娃。由于葫芦兄弟心灵相通，被囚的七弟也能为兄弟施法。

六个葫芦娃的出发位置为(0,0)，(1,0)，(2,0)，(1,1)，(2,1)，(2,2)。如果按照最快的方案，六兄弟需要多长时间才能救出七弟呢？

输入格式

输入仅包含一行，包含两个整数 X, Y，表示囚禁七弟的位置。只要把此点的六个邻接点都占了，就能把七弟救出来。 (X,Y)不会和上述六个出发点重合。0 <= X, Y < 7

输出格式

输出一行，包含一个整数，即营救的最短时间。

样例输入 1

3 1

样例输出 1

4

样例输入 2

3 4

样例输出 2

8

样例输入 3

0 5

样例输出 3

14

测试数据

共 30 组数据，输出结果近似在[0,16]内均匀分布。

注意事项

•请不要把离线计算的结果保存在源代码中（例如，直接把答案保存在 ans[X][Y]中，读取输入后直接输出），否则本题得 0 分.

•请不要特判/猜测/随机打印结果，否则本题得 0 分。

•今年新增比赛规则：公布所有选手的源代码。关于运行时限：每题均有两个时限 T1/T2。如果程序输出正确，且程序运行时间为 T，则 T<=T1 时，该测试点得分为 100%, T>=T2 时，该测试点得分为 0%。T1<T<T2 时，该测试点得分为 (T2-T)/(T2-T1) * 100%。 T1 根据参考程序的运行速度和题目难度设定，T2 通常取 T1 的 4 倍到 8 倍之间。 具体的 T1,T2 不公布。

百度之星 2009 程序设计大赛 初赛第二场试题

2009-05-31 23:19

http://www.Baiduer.com.cn 2009 年 05 月 31 日

责编:俞昊然 来源:百度爱好者

2009 年 5 月 31 日 14：00-18：00，2009 百度之星大赛在线资格赛（初赛）展开。百度爱好者（Baiduer.com.cn）在第一时间给大家带了初赛题目。第二场初赛共四题，分别是 Sorry，打错了（250 分）、树形控件（300 分）、交点覆盖（300 分）和我的地盘（350 分），总计 1200 分。

1.Sorry，打错了（250 分）

题目描述

龙先生是一位著名的记者，平时最喜欢报道一些鲜为人知的故事。最近，由于听说索马里海盗猖獗，他打算实地探访，做一个深入的调查。

龙先生联系了索马里当地的一些朋友，做了周密的计划——坐船从三亚出发，越过南海，趟过印度洋，最后到达索马里海域的亚丁湾。可就在船离海岸仅 10 公里时，突然一伙海盗突袭客船，所有人都被劫持到了索马里城内。

在人质被运送到“海盗基地”的途中，龙先生凭着多年的经验，乘海盗不注意，跃下了卡车，在无数砰砰的枪声中，没命的向外跑去。跑了几分钟后，龙先生 突然看到一个电话厅。他迅速向电话厅奔去，想打电话向住在索马里的朋友求助。然而，随着追赶脚步声的临近，龙先生估算留给自己打电话的时间最多 30 秒，

决 不容许拨错电话。

遗憾的是，越是这种危机的时候，越容易犯错。身为“智者”的您，请帮龙先生算一算：当他打电话给一位朋友的时候，恰好打给了另外一个人（不一定是他朋友）的概率是多少？已知索马里是一个有不超过 10 万人的小城，电话号码只有 6 位。

输入格式

第 1 行：索马里人数 n。n <= 100000

接下来的 n 行：所有人的电话号码

接下来的 10 行：一个 10*10 错按表，表示按 a 键时按成 b 键的概率（第一行第一列表示按 0 时按成 0 的概率，第一行第二列表示按 0 时按成 1 的概率。所有的概率用整数 0~10 表示，即实际概率要除以 10）

下一行：索马里城中你的朋友人数 m。m <= n。

接下来的 m 行：每个朋友的电话号码。

输出格式

打给你的每一个朋友时候，打通其他人的电话的概率（每行一个概率，乘以 10^6 后用整数表示）

输入样例

5

267535

229127

693606

861879

902375

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

5

267535

229127

693606

861879

902375

输出样例

4

4

4

4

4

测试数据

共 10 个测试点，基本参数如下表：

测试点编号	n	m
1	10	5
2	50	25

3	300	100
4	2000	1000
5	8000	3000
6	30000	10000
7	50000	10000
8	100000	20000
9	100000	50000
10	100000	100000

2. 树形控件 （300 分）

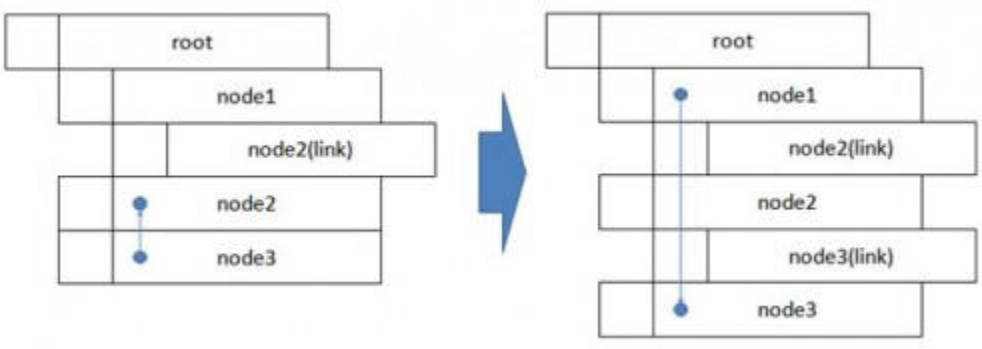
题目描述

在本题中，你需要和一种通用的交互控件——树形控件打交道。树形控件中的结点分为两类：实体结点和链接结点，后者类似于 Linux 中的符号链接。

用户可以通过拖拽节点改变树的结构。用户在一个结点 A 上按下鼠标左键，移动到另外一个结点 B 上，松开鼠标，即可将结点 A 移动到结点 B 下面，作为 B 的 最下方的儿子。如果改用鼠标右键拖动，表示结点 A 的位置不变，只是在结点 B 下创建一个到源结点 A 的链接结点。树中的所有实体节点始终是全部展开的，而链接 结点只显示结点本身，不展开被链接的子树。

屏幕是一个矩形的区域，左上角是坐标原点(0,0)，x 轴正方向指向右， y 轴正方向指向下。树的每个结点的高度为 10 个像素，宽度为 50 个像素。子结点相对父结点要向右缩进 10 个像素。另外，每个结点前都有一个展开/折叠的控 件，宽度和高度均为 10 像素。为了简单起见，这里所有的展开/折叠控件均处于失效状态，不响应与此相关的用户事件。换句话说，要想把结点 A 拖拽到结点 B 下，点击和释放时，鼠标指针必须分别位于结点 A/结点 B 所属的 10*50 区域中（不能位于该矩形的四个边界上）。如果点击或释放时鼠标指针不在任何结点区 域内，或者点击与释放时鼠标在同一个结点区域内，此操作应被认为是无效的（状态码为 1）。

即使控件能顺利检测到点击和释放时鼠标所处的结点 A 和结点 B，拖拽操作也可能是非法的（状态码为 2）。下图展示了两个连续的链接操作，其中第两个操 作是非法的，因为这将导致树上形成一个环。其他非法操作包括：将任意结点拖拽到其后代结点上（不管是移动还是链接）、尝试把任意结点拖拽到链接结点上。



如果操作完整，并且没有出现上述任何一种非法情况，则视为操作合法，状态码为 0。注意：可以将一个结点移动到其父结点下，尽管树结构不会发生任何变化。

请编写控件交互逻辑，输出每个用户操作的状态码。

输入格式

第 1 行是两个正整数 m，n，其中 m 为树的非叶结点个数，n 为操作次数。 1 <= m,n <= 1000

以下 m 行是初始树的描述。每行的开头是一个非叶结点的名称，接下来是它的各个子结点名称。每个实体节点的名称都是唯一的，是一个由数字和小写字母组 成的字符串（长度不超过 20）。如果一个结点是链接节点，其名称与源结点相同，其后用“(link)”标注（如果多次创建链接，最后也只有一个 “(link)”而不是多个）。输入保证合法（没有环，链接结点没有子结点）。在任何时候，树上最多有 10000 个结点。

之后的 n 行描述交互动作，每行格式为：

Button x1 y1 x2 y2

例如，L 15 65 25 55 表示在(15,65)单击左键并拖拽到(25,55)； R 15 65 25 55 表示在(15,65)单击右键并拖拽到(25,55)；

输出格式

输出为 n 行，依次为每个输出的返回码。

样例输入

2 3

root node1 node2 node3

node1 node2(link)

L 25 25 25 15

R 25 45 25 35

R 25 15 25 55

样例输出

1

0

2

测试数据

共 20 个测试点，基本参数如下表：

测试点编号	m	n	备注
1	1	4	
2	1	2	
3	1	3	
4	3	6	
5	1	999	
6	2	4	
7	100	100	
8	500	10	
9	500	10	
10	500	100	
11	100	300	
12	150	400	
13	200	450	
14	250	500	
15	300	550	
16	400	600	
17	600	600	
18	800	800	
19	900	900	

20	1000	1000

3. 交点覆盖 （300 分）

题目描述

平面上有 N 条直线，至少有两条直线不相互平行。

任务一：求一个周长最小的凸多边形，包围住所有直线的交点。

任务二：求一个周长最小的矩形，包围住所有直线的交点。

输入格式

第 1 行包含一个整数 T，即任务编号（T=1 或 2）。

第 2 行为一个整数 N，为直线的数目。接下来有 N 行，每行包含四个数，为一条直线上两点的坐标。

仅一行，为周长最小的矩形的周长，保留两位小数。

输出格式

如果 T=1，输出凸多边形的最小周长，保留两位小数；

如果 T=2，输出矩形的最小周长，保留两位小数。

样例输入 1

```
1

4

0 0 1 0

0 0 0 1

1 1 1 0

1 1 0 1
```

样例输出 1

```
4.00
```

样例输入 2

2

3

0 2 3 0

3 0 3 2

3 1 0 2

样例输出 2

8.85

样例输入 3

2

6

2 0 0 9

6 5 3 8

9 0 7 1

9 4 0 3

6 2 2 6

0 5 2 8

样例输出 3

51.47

测试数据

共 40 个测试点，基本参数如下表：

测试点编号	T	N
1	1	3
2	1	10
3	1	20

4	1	100
5	1	1000
6	1	2000
7	1	5000
8	1	10000
9	1	20000
10	1	50000
11-20	1	100000
21	2	3
22	2	10
23	2	20
24	2	100
25	2	1000
26	2	2000
27	2	5000
28	2	10000
29	2	20000
30	2	50000
31-40	2	100000

在所有数据中，坐标绝对值均不超过 1,000。

4. 我的地盘 （350 分）

题目描述

百度公司的员工们在工作之余，经常以产品组为单位组织一些活动，包括吃大餐、春游秋游、公益活动、唱 KTV、看电影、体育比赛等。这些活动有一个专业的名字，叫做 team building，我们也亲切的称之为“bui”。

最近，地图产品组刚刚完成一个大项目，大家决定大 bui 一场。一阵七嘴八舌后，很多内容被提了出来，最终确定先打乒乓球，然后吃饭，最后 K 歌。问题是，谁也不知道有什么地方可以同时满足这三个需求。

不过没有什么问题可以难倒我们的工程师。很快，就有人写出了程序，为大家找到了合适的地点。

地图覆盖之处，皆为我的地盘。你想挑战一下我们的工程师吗？想为我们找出更合适的地点吗？那就来吧。

输入格式

第 1 行是一个整数 k，表示某范围内所有的 POI（Point of Interest）点数量，后续 k 行每行用 5 个字段描述一个 POI 点。它们的含义和格式如下表：

内容 数据格式 数据范围

POI 编号 Int，唯一 [0,231-1]

POI 类型 字符串 0-16 字节(不超过 15 种类型)

POI 级别 Int 0-5(越大表示越高级)

POI 经度 Double [0,180],6 位有效精度

POI 纬度 Double [0,180],6 位有效精度

需要注意的是这里的经纬度跟通常的经纬度范围是不一样的

随后是一个整数 n(0 < n <= 20)，表示共 n 组查询。以下 n 行，每行表示一组查询，格式为：

POI 类型 1 POI 类型 2 POI 类型 3 最低级别 最高级别

分别表示三个 bui 地点各自的类型、最低级别和最高级别。

输出格式

对于每组查询”t1 t2 t3 min max”，输出三个 POI 编号 p1、p2、p3，满足：

•p1、p2、p3 的类型分别为 t1、t2 和 t3。

•p1、p2、p3 的级别不小于 min，不大于 max。

•p1、p2、p3 的两两欧几里得距离之和应尽量小。

输入数据保证至少存在一个解。

样例输入

5

1 休闲娱乐 3 11.122843 12.431021

2 餐饮服务 2 13.384021 10.230425

3 旅游景点 3 12.234492 9.234268

4 休闲娱乐 5 20.242391 39.304233

5 教育机构 1 42.243292 67.232065

1

休闲娱乐 餐饮服务 旅游景点 0 5

样例输出

1 2 3

测试数据

点击此处下载一份 POI 数据。所有测试点中的 POI 数据都基于此数据生成。可能的变动包括：

•修改 POI 编号

•对经纬度加入随机干扰（变化不会超过 1%）。

•修改 POI 类别和级别

•加入不超过 1%的新点，各项属性均完全随机

共 20 组测试点，其中第 i 个测试点包含 i 组查询。1 <= i <= 20

注意事项

•对于每个测试点，设已知最优解为 D，则不超过 1.05D 的任意解均是可以接受的。

•请不要把离线计算的结果保存在源代码中（例如，直接把某些输入的答案保存在常量数组中，读取输入后直接输出），否则本题得 0 分。

•今年新增比赛规则：公布所有选手的源代码。

2009 百度之星大赛在线晋级赛（复赛）

1. 高频 Query 的识别 (100 分)

内存限制:1MB

题目描述

百度每天都会接受数亿的查询请求，如何在这么多的查询(Query)中找出高频的 Query 是一个不小的挑战. 而你的任务则更加艰巨，你需要在极其有限的资源下来找出这些高频的 Query.(使用内存不得多于 1MB，本题的规定覆盖其它地方的规定)

关于内存限制：我们评测程序计算内存使用量的方法是将选手程序实际使用的内存减去以下空程序实际使用的内存.

```
#include <cstdio>

#include <iostream>

#include <string>

int main() { }
```

注意,测试的机器是 64 位的.在测试机上下面代码输出：8 8 4 2

```
#include <stdio.h>

int main() {

printf(”%d %d %d %d\n”, sizeof(long), sizeof(int*), sizeof(int),
sizeof(short));

}
```

关于时间限制：下面是一个能得到正确输出的程序，你的程序使用的时间不能大于以下程序的运行时间(对相同的输入，g++ -O2 编译).

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>
```

```
int main() {

const int L

char buf[L];

char out[

while(scanf(”%s”, buf)==1) {

int p

for(int i

p

strcpy(out[p]

}

for(int i

if(strlen(out[i])

printf(”%s\n”, out[i]);

}
```

输入格式

一行一个Query, 以文件结束符结尾。每个Query字节数L(一个汉字两个字节)满足: 0<L<=16. 输入大小不超过 1GB（包括换行符）。

输出格式

你认为最高频的 100 个 query. 每行一个, 不能有重复, 不能多输出, 但可以少输出(见样例).

样例输入

美女

帅哥

美女

百度

美女

百度

百度

美女

美女

美女

样例输出

百度

美女

评分方法

如果你的程序运行超时或使用内存峰值超过限制,那么你的得分为 0. 否则得分非 0. 你的得分是你输出的 query 的实际频次的总和(在样例中为 3+6=9)在所有有提交的选手中的排序而定.

具体来说, 设测试点分数为 S, 得分非 0 的程序数为 M ，比程序 i 的方案严格更优(实际频次的总和更大)的程序数为 Y(i) ，则该测试点程序 i 的分值为 S(1-Y(i)/M) 。换句话说，输出该测试点最优解的程序将获得 S 分，而最差解惟一的情况，输出最差解（但合法）的选手将得到 S/M 分。注意：每个测试点的得分不必为整数。

提示，请使用 C 语言的 stdio 函数而不要使用 iostream，否则在 I/O 速度上会处于明显劣势。

2. 图形检索 (100 分)

时间限制:10 秒(12:09 更新)

题目描述

和人类一样,度度熊也喜欢上网搜美女的图片,不过和人类不同,他搜的是熊熊。 他经常发现现有的搜索功能无法满足他的要求。看到喜欢的熊熊就喜欢狂搜那个熊熊的其他图片。遗憾的是人类并没有给每个熊起个名字，他非常羡慕人类可以用“金泰熙生活照”这样精确的 Query 来进行图片搜索。 有一天他终于受不了了，决定开发一个“度度熊”图片检索系统。 目的就是从一张图片出发， 检索和该图片相似的图片。

这可不是一件容易的事情，度度熊心里当然很清楚。因此他要先实现一个简化版的检索系统. 具体的描述如下:

系统目标: 给定任意一张包含特定图形的位图，从一列候选集中，正确检索出与之相似的结果.

检索对象: 黑白位图(即 0/1 bitmap)，尺寸统一为 180*180，其中每一个点 Px,y 保存像素点的颜色，1 表示黑色，0 表示白色。

输入格式

只有一个测试点，共 X=150 幅图像，分为 M=15 个类别，每个类别 N=10 个样本。在人眼看来，每个类别中的样本两两相似，但任意两个不同类别的图像都不相似。

每幅图像(按照输入顺序依次编号为 0..X-1)包含 180 行，每行 180 个字符(为 0 或 1)，字符之间无空白. 不同图像间用单个空行隔开.没有多余的输入(即保证只有 X 幅图像)

输出格式

共 X 行，其中第 i 行用 10 个整数描述图像 i 的相似图像的序号，按照相似度降序排列（即：越接近的图像越早出现）。

评分方法

对输入的每幅图像 i，评测程序将给出一个原始得分 R(i)，然后计算出所有图像的总原始得分 R=R(0)+R(1)+…+R(X-1)。 R 越大，你的最终得分也越高，但具体分数还取决于其他选手的表现。

具体来说，设原始分数非 0 的程序数为 M ，比程序 i 严格更优(总原始得分 R 更大)的程序数为 Y(i)，则程序 i 的最终得分为本题总分的 100(1-Y(i)/M)%。换句话说，原始分数最大的程序将获得本题 100% 的分数，而最差解惟一的情况，输出最差解（但合法）的选手将得到 100/M%的分数。注意：本题的最终得分不必为整数。

第 i 幅图像的原始得分 R(i)在很大程度上取决于检索出的正确图像个数 C。具体来说，当 C=0 时，R(i)=0，否则(C-1)2 < R(i) <= C2。当 C 相同时，设正确图像在输出序列中的编号分别为 T0,T1,…,TC-1，则 T=T0+T1+…+TC-1 越小越好（这里不提供具体公式）。

测试数据

本题只有一组输入. 具体见”输入格式”的描述.

共 12 类别*6 样本 = 72 幅图像, 以及相应的 0/1 矩阵的输入文件 sample.in，你可以使用该数据调试和测试你的程序. 压缩包里有一个 typerecord.txt 文件. 格式如下

0 1 13

表示 13.bmp 是第 0 个类别的第 1 个样本.

请注意样例数据和测试数据的图片个数是不同的.

3. 网页的相似度计算 (100 分)

时限:1 秒/CASE

题目描述

度度熊最近发现日本的互联网有很多恶意的作弊者用程序制造了大量的垃圾站点. 分布在成千上万的主域上, 这些站点初看起来感觉还行, 但当度度熊发现几千个格式一模一样的站点, 变化的只是其中的垃圾内容时, 度度熊觉得实在作呕. 例如下面这两个网站首页

他决定开发出一个工具, 这个工具可以用于自动比较两个网页间格式的相似程度.

网页的 HTML 标签信息组成了一棵(有根的)DOM 树. TAG 名就是树的结点的标签. 下面是一个例子:

<HTML>

<BODY>

<table>

<tr>

<td><img></img></td>

</tr>

<tr>

<td>

Welcome!

</br>

<a>Click Here!

</td>

<td><img></img></td>

</tr>

</table>

</BODY>

</HTML>

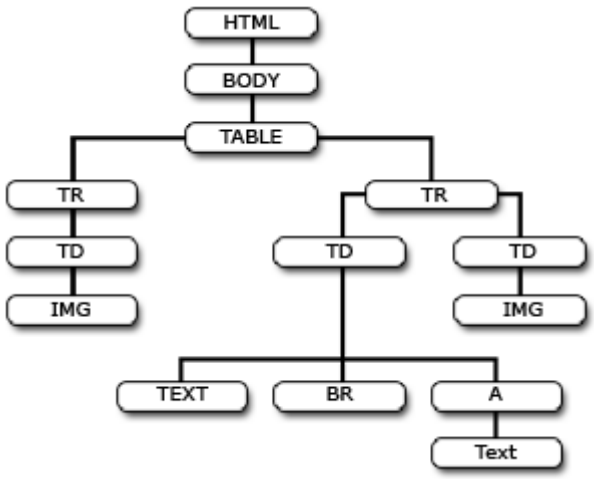


图 1

一棵”正则”的树是指结点数>1 的树.

两个 DOM 树相等当且仅当根的标签相等且各个儿子结点对应的子 DOM 树也都对应相等(各兄弟结点之间的顺序是重要的)

一个树的子树是指去掉某一结点与其父亲的边后留下来的以该结点为根的树.

如上图中以 TABLE 结点为根的子树是

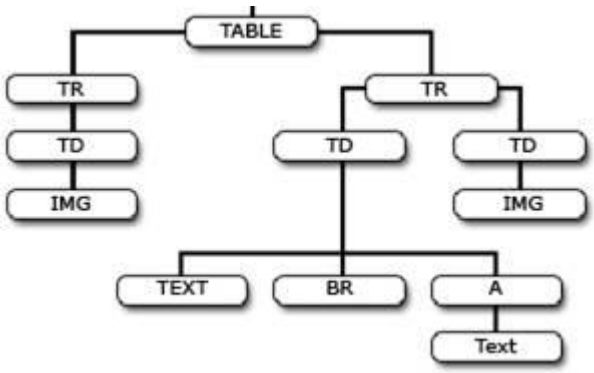


图 2

一个树的”根余子树”是指去掉一系列的子树, 但不去掉子树的根后余下的树.

例如图 1 去掉 2 个以 TR 为根的子树后的”根余子树”是:

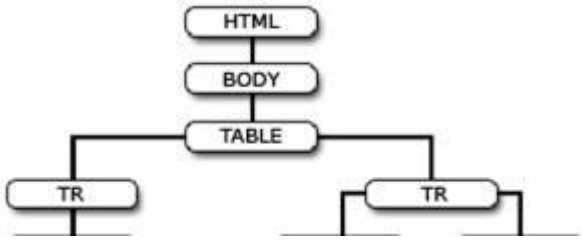


图 3

一个树的”广义子树”是指某个子树的”根余子树”, 例如图 2 子树的一个”根余子树”如下:

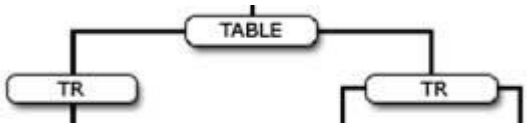


图 4

两个 DOM 树的相似度定义为这两个 DOM 树所拥有的共同的, ”正则”的 ”广义子树”的数目.

你的任务是给出 N 个 HTML 网页, 求出这 N 个网页的两两相似度(共 N*(N-1)/2 对). 并按相似度从大到小排序输出.

由于内容信息是不重要的.故你可以把连续的文本当作一个单独的虚拟 PURE_TEXT 标签.准确来说,你必须将

 why why

看作是

<a> <PURE_TEXT> </PURE_TEXT> .

但要注意.连续的空白(空格,TAB,回车换行)不是 TEXT,空白只起到分隔符的作用. 如果网页中有 PURE_TEXT 标签,则和上述虚拟的 PURE_TEXT 标签同等对待.

输入格式

第一行 N,表示有 N 个 HTML. (2<=N<=10)

从第二行开始描述 N 个 HTML,每个 HTML 的描述格式是

第一行 URL,以 http://开头。每个 HTML 的 URL 均不相同.

以下为 HTML 源码,直到一个以只有 4 个等号(====)的行表示结束.

为简单计,HTML 里的标签只有名字,没有属性信息, 并且所有的标签都是成对出现的.如
</br>.

标签名字不区分大小写(且不一定是标准的 HTML 标签).
和
是相同标签.

用正则表达式表示如下,括号表示分组:

ID:=[a-zA-Z0-9]+

TEXT:=[a-zA-Z0-9<SPACE>]+

START_TAG:=<ID>

END_TAG:=</ID>

TREE:=EMPTY|TEXT|(START_TAG TREE END_TAG)

HTML:=<html>TREE</html>

其中<SPACE>指空白(空格,TAB,回车换行等等, 具体的说, 是 c 语言中 isspace(c)为真的字符). 空白起到划分语法边界的作用, 因此可以出现在任意语法边界上. 如< / pre >是合法的.但< / p re>则不是合法的. 输入的 HTML 保证合法, 我们不会刻意出数据来测试选手对语法的理解(数据都是由真实的网页数据而来). 但选手最好作一些简单的容错. 如果我们的数据不严格符合上面的正则表达式. 但是 95%的选手程序都会正确处理, 我们将认为数据是正常的. 比赛规则中保证输入一定为 LINUX 格式, 但此题不保证.

输入文件长度<=10MB. 总的标签数目（包括 PURE_TEXT）<=10000.

输出格式

共 N*(N-1)/2 行.

每行格式为:

相似度<空格>对应的 URL1<空格>对应的 URL2

在同一行中，URL1 和 URL2 按字典序排序(即 URL1<URL2).

全部结果按相似度从大到小排序. 相似度相同的按 URL1 的字典序排序. 相似度保证小于 64 位有符号整数的最大值.

注意:我们的测试机器是 64 位, 故 long 可以保存得下相似度的结果. 请用 scanf(”%ld”,&x)或 cin>>x 来读取 long

样例输入

2

http://url_b/

<html>

<A>TEXT
T</br>TEXT

</html>

====

http://url_a/

<html>

<a>U
U</br>

</html>

====

样例输出

2 http://url_a/ http://url_b/

测试数据

共 10 组测试数据，测试数据是真实的网页数据(经过必要的转换以满足上述的格式描述和规模)加部分手工数据。

4. 拼车 (100 分)

时间限制:10 秒

题目描述

虽然北京的公交系统很发达，但对上班一族来说，出租车仍是很常用的交通工具. 度度熊就经常坐出租车上下班. 有一天他发现一个现象：他的同事基本上都是每人单独坐一辆车，虽然有些人住在同一个方向，甚至同一个小区。他想，可以考虑让大家拼车，即节约出租车资源，又可以减少大家的支出.

度度熊作了以下假定:

出租车最多一辆可以坐 4 个人(为简单计,不考虑像度度熊这样因体积太大而坐不了 4 个人的情况).

假设公路都是横平竖直的，可以用正方形框格来表示公路。相交的公路视为连通（可以在交叉点处从一条公路开到另一公路上）。车只可以在公路上走。

员工家都在公路的交叉点处。员工下班后，都从公司所在位置直接打的士回家。

每位员工只能乘坐一辆出租车到达目的地，不能中途下车，换乘其他出租车。

同往常一样，度度熊只喜欢出主意，却不喜欢写代码．所以这个重任就只好落到你身上了:）

你的任务是写一个程序：求出一种员工打车方案，使得把所有员工送回家，在这种情况下，要使所有的士所走的路程尽量短。

为了方便起见，用笛卡尔坐标系来描述员工家所在区域。假定员工家所在区域的长和宽均为 9。

下面是公司所在位置为（3,3）时，坐标描述图：

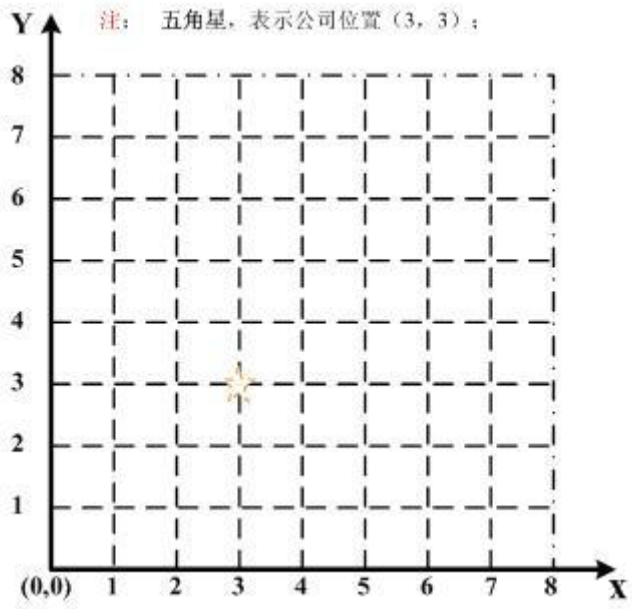


图 1．公路示意图.

输入格式

输入包括：公司位置，员工总人数 和 每个员工家所在的位置。

具体格式见下：

第 1 行：有 1 个数 N（0<N<100），表示当天需要打的回家的员工总人数。

第 2 至 N+1 行，每行有 2 个整数，分别为 Xi（0<=Xi<=8）和 Yi（0<=Yi<=8），中间用空格隔开，表示员工 i 所要到达目的地的坐标。例如，第 2 行是员工 1 的家的坐标位置，第 3 行是员工 2 家的坐标位置，依次类推。不同员工的住址可能完全相同，员工的住址也可以和公司重合（可以在程序中直接忽略他）。

第 N+2 行有 2 个数，分别为 M（0<=M<=8）和 W（0<=W<=8），中间用空格隔开，表示公司的坐标。

输出格式

输出包括：总的最短公里数，所需车的数目，以及每辆车所载的员工（用坐标表示）和每辆车所走的路线（用坐标表示）。

具体格式见下：

第 1 行：总的最短公里数。

第 2 行：所需车的总数目，假定是 C；

下面第 3 行至第 C+2 行，每一行输出包含三部分：

第一部分是该车所走路程；

第二部分是该车所载的员工，目的地坐标表示；

第三部分是该车所走的路线，用经过的交叉点坐标序列表示。

第一部分和第二部分之间用 空格 分割；

第二部分和第三部分之间用“:”分割。

样例输入

8

0 0

4 0

7 1

5 4

1 5

3 6

6 6

1 8

3 3

样例输出

29

4

6 (0,0) : (3,3) (2,3) (1,3) (0,3) (0,2) (0,1) (0,0)

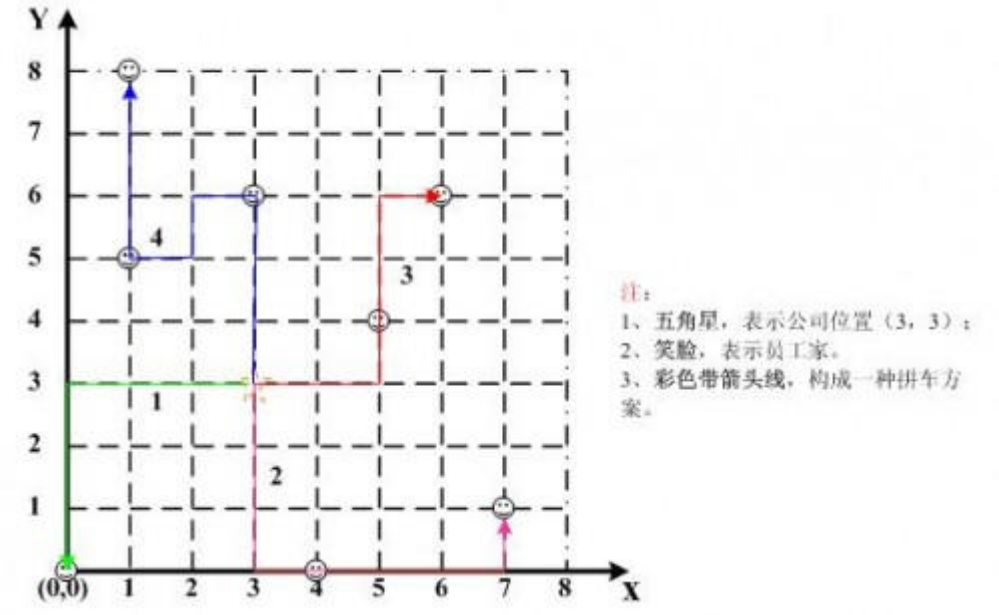
8 (4,0) (7,1) : (3,3) (3,2) (3,1) (3,0) (4,0) (5,0) (6,0) (7,0) (7,1)

6 (5,4) (6,6) : (3,3) (4,3) (5,3) (5,4) (5,5) (5,6) (6,6)

9 (1,8) (1,5) (3,6) : (3,3) (3,4) (3,5) (3,6) (2,6) (2,5) (1,5) (1,6) (1,7) (1,8)

样例解释

输入的员工家、公司具体位置以及拼车方案见下图:



此例的一个打车方案为:

1 号线路程为 6 公里,载 1 人。
2 号线路程为 8 公里,载 2 人。

3 号线路程为 6 公里，载 2 人。
4 号线路程为 9 公里，载 3 人。
总计最短路程为 29 公里。

评分方法

如果你的程序输出方案非法，那么你的得分为 0. 否则得分非 0.你的得分是由你的方案得到的总的最短公里数在所有有提交且得分非 0 的选手中的排序而定.

具体来说，设测试点分数为 S，得分非 0 的程序数为 M ，比程序 i 的方案严格更优(总的公里数更小)的程序数为 Y(i) ，则该测试点程序 i 的分值为 S(1-Y(i)/M)。换句话说，输出该测试点最优解的程序将获得 S 分，而最差解惟一的情况，输出最差解（但合法）的选手将得到 S/M 分。注意：每个测试点的得分不必为整数。

测试数据

共 25 组，员工人数近似在[0,100]中均匀分布，公司位置和每个员工家所在位置都是随机生成的。

百度之星 2010 程序设计大赛 决赛试题

http://www.Baiduer.com.cn 2010 年 07 月 28 日 责编:刘潇磊 来源:百度爱好者

百度爱好者（Baiduer.com.cn）消息，百度之星 2010 程序设计大赛决赛 7 月 27 日早 9 点 30 分在百度大厦五福降中天会议室鸣锣开战。百度爱好者给大家带了决赛题目，供有兴趣的朋友研究。试题以“僵尸大战植物”为主题。

题目描述

这是植物大战僵尸里面的一款小游戏《我是僵尸》endless 模式，根据植物布局，我们合理的安排僵尸攻击植物，直到吃光每行最左边的大脑。每派出一个僵尸，将会花费你的太阳（购买僵尸的货币），我们目标是就是用最少的太阳通过给定的一组关卡。

题目简化和细节说明

植物的脸都朝右，僵尸脸都朝左，僵尸是从最右面攻过来的；

僵尸不是慢悠悠走的，而是一次走一格或者两格；

僵尸前面有植物，必须吃完植物后才能前进，除非植物被梯子僵尸架了个梯子；

僵尸跟植物（大脑）在同个格子的时候，僵尸才可以吃植物（大脑）；

大脑的血量为 1， 僵尸咬一口就吃掉。吃完大脑后，僵尸消失；

地图上布满了植物（没有空位），僵尸从地图最右边开始攻击；

一个僵尸只会攻击同行的植物，而且永远在这一行上；

没有特殊说明，植物的攻击都只对同一行上的僵尸有效；

有些植物会攻击旁边行的僵尸，如磁铁蘑菇（请参考磁铁蘑菇的说明）；

有些植物会受到旁边行的僵尸影响，比如胆小蘑菇（请参考胆小蘑菇的说明）；

铁桶僵尸，橄榄球僵尸和梯子僵尸在被植物攻击的时候由铁桶，铁帽子，梯子先承受攻击，最后才轮到僵尸本体承受攻击。如果周围有磁铁蘑菇，这三样都会被磁铁蘑菇吸走，被吸走后就是本体直接被攻击。吸走后的铁器将消失；

植物选中一格攻击，发现该格有很多僵尸，而自己只能攻击一个，没有特殊说明，选择先放入的僵尸。如果这些僵尸是同一时刻放入的，选择先出现在输入文件的僵尸。

模拟器细节

我们模拟器分，植物，僵尸，子弹来设计。植物攻击僵尸有几种方式

1、直接攻击僵尸，比如窝瓜，地雷，食人花，磁铁蘑菇等，这种靠自身能力攻击僵尸，在攻击的瞬间就会对僵尸造成伤害。

2、选择发射子弹，发射子弹的行为是不会让僵尸立即扣血，而是等到判断子弹是否能够打击到僵尸时，才会扣僵尸的血。

我们的模拟器将地图变成一个 Array[Row][Col]的矩阵，僵尸的行走速度是 1/s 或者 2/s 只会从 Array[Row][i]走到 Array[Row][i-1]或者 Array[Row][i-2]。速度为 2 的僵尸，如果前面有植物挡住，只会前进一格；同理子弹的也是一样。

其它模拟细节

1、这个游戏每 1 秒检测一次，生成该帧的状态图，然后开始判断；

2、每帧的状态生成之后，判断伤害顺序为；

1、检查每个僵尸吃植物（大脑）；

2、检查每个植物是否要攻击僵尸（直接攻击，或者发射炮弹）；

3、检查每枚炮弹是否打击到僵尸；

3、僵尸吃植物，植物死亡时，将立即从植物列表删除；同理炮弹打到僵尸，或者僵尸死亡，也都会立即从对应的链表中删除。极端例子：当植物和僵尸都只剩一滴血的时候，由于僵尸先咬植物，所以植物死了，僵尸不被攻击；

4、植物发射炮弹的时候，炮弹的起始位置为植物所在位置，下一秒开始移动。

植物数据

植物的伤害，攻击性弹药碰到僵尸后都造成 1 的伤害。

植物	特性	血	代号
 土豆地雷	• 能够给予僵尸致命一击，僵尸碰到土豆雷后，本格僵尸全死. • 土豆地雷爆炸后消失	无穷	**Potato-Mine**
 食人花	• 食人花会吞掉他前面1格或本格的僵尸，优先吃本格的 • 在咀嚼僵尸的时候，食人花不能动弹，咀嚼僵尸需要花费40秒的时间 • 咀嚼的时候,食人花被杀死,肚子里的僵尸不会复活	12	**Chomper**
 豌豆射手	• 能够发射豌豆把僵尸干掉，一发豌豆会消耗僵尸1格血,不能穿透僵尸. • 只有发射的豌豆消失之后，才能发射下一枚 • 发射速度,每秒3格,打中僵尸后消失，超出地图范围也消失 • 豌豆只有发现自己前面有僵尸才会开火（同行）	12	**Peashooter**
 坚果	• 坚果具备坚硬的外壳，可以用来挡住其它僵尸. • 没有攻击能力	140	**Wall-Nut**
 喷射蘑菇	• 僵尸（同行，蘑菇前面）靠近喷射蘑菇3格以内，喷射蘑菇才会开火 • 喷雾只有攻击到僵尸或者超出地图范围才消失 • 发送速度每秒3格	12	**Puff**
	• 靠近窝瓜的僵尸会被压扁,消灭该格上的所有僵尸.	无穷	**Squash**

僵尸数据

僵尸吃植物每秒 2 格血。

僵尸	特性	速度	血	花费	代号
 小鬼僵尸	移动速度快	2 格/s	3	50太阳	**Imp**
 路障僵尸	很普通	1 格/s	29	75太阳	**Conehead**
 铁桶僵尸	优先攻击铁桶，铁桶脱落后攻击本体 铁桶受到一定次数攻击脱落或者被磁铁蘑菇吸走	l 格/s	铁桶 能够承受56次攻击 去掉铁桶后能承受10次攻击 总计66	125太阳	**Bucket**
 橄榄球僵尸	优先攻击铁帽，铁帽脱落后攻击本体 铁帽受到一定次数攻击脱落或者被磁铁蘑菇吸走	2 格/s	铁帽子能够承受70次攻击 去掉铁帽子之后僵尸承受12次攻击 总计82	175太阳	**Football**
 梯子僵尸	碰到第一个植物，将梯子放在上面，下一帧计算才会爬过去。 植物被架了梯子之后，会被僵尸直接越过 无论梯子在僵尸手里，还是在植物身上，都会被磁铁蘑菇吸走。 只有僵尸手上没有梯子，才会攻击到僵尸本体 梯子受到一定次数攻击后会脱落，消失。 植物被架了梯子，只有植物死了或者被磁铁蘑菇吸走，梯子才会消失。	有梯子时2格/s， 没有梯子时1格/s	梯子能够承受24次攻击 僵尸能够承受24次攻击 总计 48	150太阳	**Ladder**
 撑杆僵尸	前进途中会翻过遇到的第一植物落到植物后面的那格上。检测到可以翻越，立即翻越不需要等到下一帧。	有杆时2格/s 没杆时1格/s	24	75太阳	**Pole**

程序输入输出

现在假设我们每个游戏场景都是在 ROW*COL 的方格中满植物，僵尸只能放置在最右面一列的方格中，并发起进攻。地图上放满植物每个方格中的可以同时存在多个僵尸。

输入：

输入由标准输入给出

第一行为 1 个整数 M（0<M<100000000），第一个表示下面这些关卡你总共只能用 M 的太阳；

第二行也是 1 个整数 N（0<N<100000），表示下面有 N 个 case；

第三行开始为 N 个 case 的具体信息，每个 case 第一行为 2 个整数 Row，Col 其中 Row 表示多少行，Col 表示多少列（1<=Row<=10000， 1<=Col<=10000）接下来一共跟着 Row 行，并用代号标识每行的植物类型， 并且用空格隔开。

输出：

结果需要输出到标准输出

对于每个关卡，第一行为一个整数 K，表示本关使用的僵尸数，接下来 k 行表示每个僵尸的布局，每行有 3 单词来表示一个僵尸的信息， 以空格隔开，分别表示 僵尸代号（ID）， 行（r）， 时间（time）。它们表示在时间 time 的时候把一个僵尸放到第 r 行的最右边上. 对于放置时间相同的僵尸，植物攻击顺序和输出中的顺序一致的。

时间是整数，从 0 开始计算。我们要求输出的行按僵尸先后放置的时间，顺序一致。

如果你无法解出本关，又不想浪费太阳，请输出 1 行 0；

如果你输出了僵尸，那么即使这个僵尸是过关之后放置的，都要计算它的花费。

在桌面上有一个模拟器，填上你的输入输出文件它可以帮忙计算出你总的花费。

评判标准

我们评判程序会运行多个输入， 然后我们会统计你的解决方案购买僵尸的总开销。

1、通过关卡最多的获胜；

2、关卡数一样， 根据开销由小到大进行排序，决定谁获胜。

如果花费购买僵尸的开销相同，那么会以总的游戏的运行时间为标准，游戏运行总时间少的程序获胜，每关游戏运行时间指顺利过关的时间而不是程序执行时间。

注意事项

1、如果僵尸放置的位置不在地图上，那这个僵尸会被认为无效的，同时相应的花费依然会被扣除；

2、如果僵尸的代号错误（注意大小写和连字符），那么这个僵尸会被认为是非法的，不进行任何操作， 同时会按照最贵的橄榄球僵尸扣除 175；

3、如果程序出现异常情况或者程序本身没有完成全部的关卡， 那么排名的时候会以通过的关卡数为准；

4、如果在某一关植物已经全部被消灭，但是还有僵尸被放置到地图上，这些僵尸的花费依然会被计算；

5、如果花费的太阳数超过数据的给定值， 模拟器在放置僵尸花费超过现在太阳数的时候结束游戏，以此时的结果作为最后结果。

【Sample Input】

100000

3

5 1

Potato-Mine

Potato-Mine

Potato-Mine

Potato-Mine

Potato-Mine

5 2

Potato-Mine Potato-Mine

Potato-Mine Potato-Mine

Potato-Mine Potato-Mine

Potato-Mine Potato-Mine

Potato-Mine Potato-Mine

5 2

Peashooter Wall-Nut

Peashooter Wall-Nut

Peashooter Wall-Nut

Peashooter Wall-Nut

Peashooter Wall-Nut

【Sample Output】

5

Pole 0 0

Pole 1 0

Pole 2 0

Pole 3 0

Pole 4 0

10

Imp 0 0

Imp 1 0

Imp 2 0

Imp 3 0

Imp 4 0

Pole 0 1

Pole 1 1

Pole 2 1

Pole 3 1

Pole 4 1

5

Pole 0 0

Pole 1 0

Pole 2 0

Pole 3 0

Pole 4 0

输入输出说明

第一关：每列都是土豆雷，我们都用撑杆跳的过去。如果你派小僵尸过去，那么僵尸会全砸死，虽然植物也全被消灭了，但是你还是没过关，还需要派小僵尸，花费更多。

百度之星 2010 程序设计大赛 复赛试题（上）

http://www.Baiduer.com.cn 2010 年 07 月 06 日 责编:张娜 来源:百度爱好者

百度爱好者（Baiduer.com.cn）消息 2010 年 6 月 19 日，2010 百度之星大赛复赛展开。百度爱好者给大家带了复赛题目，供有兴趣的朋友研究。复赛共十题，上期五期。分别是 A+B 问题、i-Doctor、url 规范化、并行修复、猜猜你在哪儿。

1.A+B 问题（时限：5000ms）

问题描述

Suzumiya 最近开始无端刁难她的同学 ViVo，总是莫名其妙的问他一些简单而又离谱、没有实际意义的数 学问题，比如三千加上五百等于多少。回答一次两次还可以，但不断这样的纠缠致使 ViVo 已经无法忍受了。所以 ViVo 决定制作一个语音装置来自动回答 Suzumiya 提出的无聊问题。

ViVo 知道你是个伟大的算法艺术家，所以希望该装置核心的数学计算模块你能够来帮忙。装置接收到的语音会自 动为你转化为对应的中文字符串，经过你的模块处理完成后输出中文字符串，传递给装置来朗读出来。

为了给你带来方便，ViVo 已经提前收集好了 Suzumiya 可能会问到的问题，发现这些问题中大部分是数学 加法，也还有一些不是加法的问题，但答案依然都可以用数字来表示。

输入格式

输入的第一行是一个数字 n，表示接下来有 n 个问题，每个问题占一 行。

提示：最终评测时所用的输入数据可以在这里(windows)和这里(linux)下载。

输出格式

每行包含一个没有语病的中文表示最终的结果。

样例输入

2

一加一等于几?

三千加上五百等于多少?

样例输出

二

三千五百

提示

请注意：不要提交你的输出文件，而应当像其他题目一样，提交你的源代码。本题的最终得分计算如下：

假设输入除第一行数字 n 外有 n 个非空行,你的输出必须也恰好包含 n 个非空行，否则本题得 0 分。

从前向后一一比较，如果你的输出有 m 行和标准答案一致，你将得到本题的 $100*(m/n)^3$%。换句话说，如果你的程序有 70%的行和标准答案一致，你将得到本题约 34.3%的分数。

2.i-Doctor（时限：3000ms）

问题描述

百度计划开发一个在线的健康专家系统,帮助用户足不出户就能初步了解自己所患的疾病，并以此向用户推荐适合的 医院就诊。经过对海量数据的分析，百度提取出了若干表征疾病性状的特征,并把每种疾病是否符合某项特征都进行了标记，最终得到如下数据表格：

疾病名称	A_0	A_1	…	A_{n-1}
D_0	Yes	Yes	…	Probably
D_1	No	Probably	…	Yes
…	…	…	…	…
D_{m-1}	Probably	No	…	Yes

其中，$D_0, D_1, \cdots, D_{m-1}$ 表 示疾病名称，$A_0, A_1, \cdots, A_{n-1}$ 表示疾病的特征。m、n 均为正整数且 $m < 4096$，$n < 128$。特征的取值为 Yes（符合该项特征）， Probably（有可能符合该项特征）或 No（不符合该项特征）。

这个专家系统被命名为 i-Doctor，因为它的工作方式很人性化，就像医院的专家一样通过与病人的一问一答 来得出诊断。每当开始一个诊断时，i-Doctor 首先提问：”你是否感觉到有 A 症状？” 其中，A 为一疾病特征。用户依据自己

的感觉回答。 不幸的是，有时候病人对自己是否有 A 症状不能肯定，甚至会给出错误的回答。统计表明，病人的回答及置信度如下：

用户回答	含义	置信度
Yes	是	0.95
Probably yes	很是	0.80
Probably no	不很是	0.80
No	否	0.95
Don't know	不知道	-

注意：每个病人在诊断之前患有一种（且仅一种）确定的疾病，且该疾病保证存在于上述疾病数据库中。

现在，请你编写一个程序来让 i-Doctor 开始工作。

交互

你的程序应当读写标准输入输出，以便与测试库交互。交互方式如下：

首先，你的程序（doctor）应从标准输入读取疾病特征表。第一行是两个正整数 m 和 n，表示疾病的种类数和特征的种类数。m 和 n 之间以一个空格 隔开。接下来共有 m 行，其中每一行描述一种疾病，格式为：

疾病名称 特征值 0 特征值 1 … 特征值 n-1

开头的字符串为疾病名称，长度不超过 7 字节；一个空格之后依次是各特征值，取值为英文字母 Y 或 N 或 P，分别表示 Yes、No 和 Probably。相邻特征 值以一个空格隔开。

接下来，你的程序可以向病人（patient）提问，提问方式为在标准输出上打印一行，格式为：Do you feel Ai? 其中 Ai 表示特征特 i。 此后，你的程序应当从标准输入读取病人的回答。病人每次的回答也为一行，内容为 Yes、Probably yes、Probably no、No 和 Don’t know 之一。

如此一问一答，直到你的程序诊断出病人所患疾病，此时应在标准输出上打印一行：I think of Dj! 其中 Dj 为此疾病名称

如果无法确诊，你的程序可以在标准输出上打印一行：Give up. 表示你放弃诊断该病人。注意：很快你将看到，放弃诊断总比错误的诊断要好。

在确诊或放弃后，你的程序应自行终止。

可以在这里 (windows)和这里 (linux)下载测试库（内附使用说明）。

程序举例

下面是一个示例程序（省略了 include），它能正确的与测试库进行交互，尽管它的得分可能不高：

```
int main()

{

int m, n;

char name[10], line[256];

char table[4096][128][2];

int i, j, q;

srand(time(0));

scanf(“%d %d\n”, &m, &n);

for(i = 0; i < m; i++)

{

scanf(“%s”, name);

for(j = 0; j < n; j++)

scanf(“%s”, table[i][j]);

fgetc(stdin); /* consume ‘\n’ */

}

for(q = 0; q < 4; q++)

{

printf(“Do you feel A%d?\n”, rand() % n);

fflush(stdout); /* Important */

fgets(line, sizeof(line), stdin);

}

if(rand() % 3 == 0)
```

```
printf(“I think of D%d!\n”, rand() % m);

else

printf(“Give up.\n”);

fflush(stdout); /* Important */

return 0;

}
```

注意,你的程序应当像上面的程序一样,在每次输出后立即执行 fflush(stdout)语句，使输出的内容立即被送入测试库（而不是留在输出缓冲区中）。另外，你的程序应能独立编译，不能依赖于任何其他外部文件 （包括测试库）。

评分方法

本题一共有 25 个测试点，每个测试点分值相同。每个测试点对应一个病人，如果你诊断错误或者放弃诊断，则得分 为 0。如果你成功诊断，你的得分取决于你询问的次数和其他选手成功诊断该病人所用的询问次数。当然，次数越少得分越高。注意：所有测试点的得分相加后，还 要扣除错误诊断的惩罚。具体来说，每次错误的诊断将扣掉两个测试点的满分(哪怕你没有任何测试点得到了满分)。如果扣除惩罚后，所得的分数为负，总分将调 整为 0。

3.url 规范化 (时限：5000ms)

问题描述

互联网上很多不同 url 都是指向同一页面的，比如：

http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1

http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1&fr=tb0_search

http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1&fr=tb0_search&ie=utf-8

都指向的是同一页面：
http://tieba.baidu.com/f?kw=%C9%CF%BA%A3%CA%C0%B2%A9%BB%E1。

我们需要把不同形式的 url 规范化，用统一的 url 代替。目前已经定义了一批规范化规则，你的任务是把一些待处理的 url 按规则替 换成新的 url。

每条规则都是一个字符串。为了方便理解，我们定义如下术语：

通配符：字符’#’或者’*’，以及紧跟其后的长度限定符（可选）。其中’#’匹配数字，’*’匹配任意字 符。

长度限定符：紧跟通配符后，格式为[min-max]，表示该通配符匹配的字符个数的最小值和最大值，其中 min 和 max 均可以省略（但不能同时省略），当 min=max 时可以简写为[min]。’#’的默认最小长度为 1，’*’为 0。二者的默认最大长度均为 无穷大。例如，”#[3]“表示 3 个连续数字，’*[3-9]‘表示 3 到 9 个任意字符，’*[-8]‘表示 0-8 个任意字符，’#[7-]‘表示至少七个 连续数字。注意，当通配符的后一个字符为左方括号’[’时，它总是应当被看作长度限定符的开始标志。

尖括号：成对的小于符号"<"和大于符号">"。尖括号总是两两配对，不存在无法配对的孤立尖括 号，且尖括号内部不会出现通配符，也不会嵌套另一对尖括号。

规则片段：在处理规则之前，我们把规则分成若干个连续的片段，每个片段是单个通配符（以及紧随其后的长度限定 符，如果有的话）、一对尖括号（以及括号内的字符，如果有的话），或者若干个连续的普通字符（不含通配符和尖括号）。为了避免歧义，规则应当被划分成最少 的片段。不难证明，此时的划分方法是惟一的。

例如，规则"abc*d#[4-7]eeef<gh>i”包含 7 个规则片段：abc、*、d、#[4-7]、eeef、<gh>、i。

用某一条规则处理 url 时，我们遵循以下步骤：

第一步：匹配。首先，忽略含有尖括号的规则片段，用其他片段来匹配 url，使得每个片段所匹配的字符串从左到右 连接之后和该 url 完全相等。注意，连接顺序必须按照规则片段的顺序从左到右进行。根据定义，由普通字符组成的片段只能匹配和它完全相同的字符串，而通配 符可以匹配多样化的字符串，规则如前所述。例如，abc<x>def*[3-5]ghi 在忽略尖括号后得到四个片段：abc、def、* [3-5]、ghi，因此，abc<x>def*[3-5]ghi 能匹配到的 url 是那些以 abcdef 开头，ghi 结尾，中间有 3 到 5 个字 符的字符串。

如果无法匹配，说明该规则不适用于此 url，处理结束；但有时会出现匹配方式不止一种的情况，还需要消除歧义。 例如*bc<d>*<xyz>匹配 abcabca 的方法有两种：

规则片段	*	bc	<d>	*	<xyz>
方案一中匹配的url片段	a	bc	无（匹配时忽略尖括号所在片段）	abca	无（匹配时忽略尖括号所在片段）
方案二中匹配的url片段	abca	bc	无（匹配时忽略尖括号所在片段）	a	无（匹配时忽略尖括号所在片段）

在本题中，如果出现像这样匹配方法不惟一的情况，你应当选择让从左到右第一个通配符匹配字符数最少的方案。如果还有多种方案，再选择其中让第二个通 配符匹配字符串最少的方案，以此类推。 在上面的例子中，应选择第一种方案。

第二步：替换。对于每对尖括号，把它左边相邻的 url 片段替换为尖括号内的 文本，然后连接所有 url 片段（输入数据保证规则不以左尖括号开头，并且相邻两对尖括号之间至少包含一个字符）。我们仍然借助上例说明问题：

规则片段	*	bc	<d>	*	<xyz>
匹配的url片段	a	bc	无（匹配时忽略尖括号所在片段）	abca	无（匹配时忽略尖括号所在片段）
替换后的url片段	a	d	无（匹配时忽略尖括 号所在片段）	xyz	无（匹配时忽略尖括号所在片段）

也就是说，”abcabca”被规则”*bc<d>*<xyz>”处理后变成了”adxyz”。

再举一个实际的例子：*/viewthread.php?tid=#&<>*<> 表示将 url 文件名为 viewthread.php，第一个参数名为 tid 且参数值为数字时，把其它参数删除。 换句话说，这条规则将把 www.baidu.com/dir/viewthread.php?tid=123&arg=aa 替换为 www.baidu.com/dir/viewthread.php?tid=123。

输入格式

第一行为一个正整数 M，说明共有 M 条规则。第二行为一个正整数 N，说明共有 N 条 url。接下来 M 行，每行代表一条规则，不超过 256 个字符。 再接下来 N 行，每行代表一条 url。所有 URL 都去掉了 http:// 头，且不含#*<>四种特殊字符，也不超过 256 个字符。M<=1000，N<=10000，输入的 url 和规则均不含汉字。

输出格式

输出共 N 行。对每一行输入的 url，按照输入顺序依次考虑所有规则，直到某个规则修改 url 时输出替换后的新 url（如果匹配成功，但修改前后 url 并没有发生变化，不算修改，应继续尝试剩下的规则），然后忽略剩下的规则。如果没有匹配的规则，则把输入的 url 按照原样重新输出。

样例输入

5

7

blog

you

club

www

blog

blog
rela_prevarticle

you

club

www

blog

www

www

样例输出

blog

you

club

www

blog

www

www

评分方法

本题按测试点评分，每个测试点的得分计算方法如下：

假设输入有 n 条 url，你的输出必须也恰好包含 n 行，否则本题得 0 分。

从前向后一一比较，如果你的输出有 m 行和标准答案一致，你将得到本题的 $100*(m/n)^3$%。换句话说，如果你的程序有 70%的行和标准答案一致，你将得到本题约 34.3%的分数。

50%的数据保证 N<=100，90%的数据保证 N<=1000。

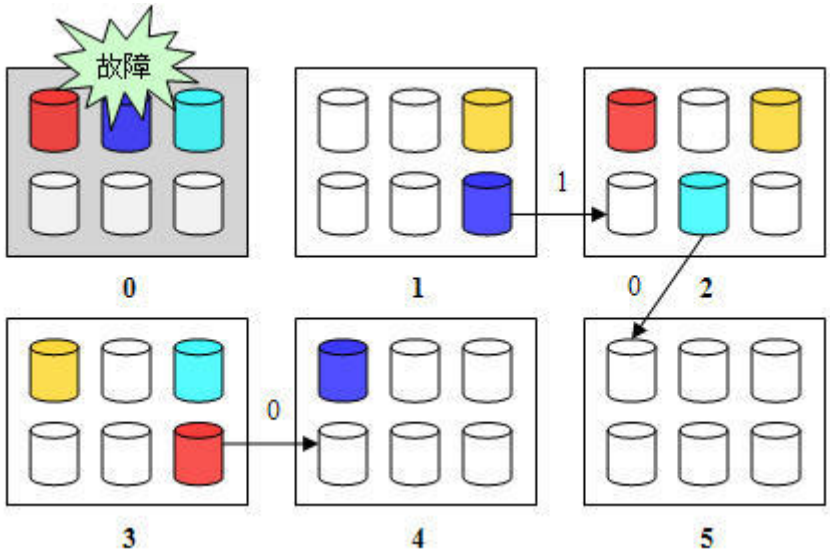
4.并行修复 （ 时限：5000ms ）

问题描述

百度的网页按照某种标准均匀划分为 N 组，每组数据编号为 0 到 N-1 的整数。一组网页可以全部存入一个数据库 中。 为了保证数据的可靠性，每一组网页按 X 份镜像存储，互为备份，且任两份镜像不会分布在同一台主机上。 当有 F 台机器因故障而失效时，部分网页组对应的镜像数目可能会低于 X 份，此时需要复制镜像，使系统重新恢复到每组网页均有 X 份镜像的状态。

假设系统中一共有 M 台主机，主机编号从 0 到 M-1。每台主机的性能和网络带宽均相同。一台主机最多可以存储 T 个大小相同的数据库。每台主机任意时刻最多允许 C 个并发网络连接。一次复制操作占用源主机和目标主机各一个网络连接，源主机和目标主机不允许相同。若一对 源主机和目标主机在同一时刻进行多次(B > 1)复制操作，则占用源主机和目标主机各 B 个网络连接。记一份镜像的一次复制用时为 1 单位。假定每次复制的启动时间点必须为非负整数，请设计一种修复算 法，对于给定的某种镜像分布状态和失效的主机编号集合，输出一种恢复方案，使数据恢复的总用时尽可能短。输入集合保证系统能够从故障中恢复。

下图中给出了当 N=4，X=3，M=6，T=6，C=1 时，0 号主机故障时的一种恢复方案，箭头表示复制的方 向，箭头盘旁的数字代表该次复制的启动时间点。



输入格式

输入包含 N+2 行。第一行包含五个正整数 N，X，M，T，C，接下来的 N 行依次表示第 0 到 N-1 组网页的镜像 分布状态，每行有 X 列整数，为该组网页的镜像所在的主机编号。 最后一行由 F 个整数组成，表示故障主机的编号（注意：F 本身不在输入中出现，你需要统计最后一行的整数个数，以获取 F 的值）。N<=40000，2<=X<=8，2<=M<=4000，T<=600，C<=4，F<=10。

输出格式

输出一个数据恢复方案。方案包含若干行，每行代表一次镜像复制操作，由四个整数 W，S，D，T 组成。W 代表网页组编号，S 为源镜像所在主机编号，D 为目标

镜像所在主机编号，T 为复制的启动时间点（起始时间为 0）。各复制操作应按照 T 从小到大排 序。

样例输入

4 3 6 6 1

0 2 3

0 1 4

0 2 3

1 2 3

0

样例输出

0 3 4 0

2 2 5 0

1 1 2 1

样例说明

该方案用了 2 单位时间（耗时最多的主机编号为 2）。

评分方法

本题采用相对评分。对于每个测试点，你的最终得分取决于你输出的方案和其他选手输出的方案的相对优劣程度。

5.猜猜你在哪儿（ 时限：1000ms ）

问题描述

有一个半径为 1 米的圆，圆心位于(0,0)点。你和圆处在同一平面上，准备和这个圆一起玩游戏。这个圆每过一 分钟会随机移动一次，一共移动 t 次。 圆有一个移动距离限制参数 s，每次圆心从(x1,y1)移动到(x2,y2)时总是满足：
|x1-x2|<=s, |y1-y2|<=s。

每次圆移动完成后，你可以依次询问 k 个点是否在圆内，然后任意走到一个新的位置，结束这个 回合。你的目标是尽量让自己位于圆内，并且离圆心越近越好。每个回合结束后，若你在圆的边界上或者圆外，得 0 分；若在圆内，得分为(1-

你到圆心的距离/圆的半径)。所有 t 个回合结束后，每个回合的平均得分就是你对于该测试点的原始得分。

交

你的程序应当读写标准输入输出，以便与测试库交互。交互方式如下：

首先，你的程序应从标准输入读入三个数 t, s, k，分别表示总回合数、圆心移动的距离限制，以及每回合你可以询问的点数。1<=t<=100, 0 < s < 1, 1<=k<=10。

接下来，你应当分 k 行给出各个询问点的坐标——每行两个数 x 和 y，即询问坐标为(x,y)的点。每次询问一个点之后，你的程序应当从标准输入读入 一个数 a，1 表示询问点在圆上或者圆的边界上，0 表示在圆外。

再接下来，你的程序应当往标准输出写两个数 x 和 y，表示你要走到的新位置的坐标。

完成所有 t 个回合之后，你的程序应当自行退出，否则将按超时处理。

可以在这 里(windows)和这里 (linux)下载测试库（内附使用说明）。

程序举例

下面是一个示例程序（省略了 include），它能正确的与测试库进行交互，尽管它的得分可能不高：

```
int main()

{

int t, k;

float s;

scanf(“%d%f%d”, &t, &s, &k);

for(int i = 0; i < t; i++)

{

for(int j = 0; j < k; j++)

{

int ans = 0;
```

```
printf(“0.%d 0.%d\n”, rand(), rand());

fflush(stdout);

scanf(“%d”, &ans); // 1 表示仍然在圆里或圆的边界上，0 表示在圆外

}

printf(“0.%d 0.%d\n”, rand(), rand());

}

}
```

注意，你的程序应当像上面的程序一样，在每次读取新的输入之前调用 fflush(stdout)，以确保这之前输出的内容立即被送入测试库（而不是留在输出缓冲区中）。另外，你的程序应能独立编译，不能依赖于任何其他外 部文件（包括测试库）。

评分方法

本题采用相对评分。对于每个测试点，你的最终得分取决于你的原始得分和其他选手的原始得分的相对优劣程度。

2011 年百度之星程序设计大赛试题

初赛题目

第一场

一．图标排列

描述

百度应用平台上有很多有趣的应用，每个应用都由一个开发者开发，每个开发者可能开发一个或多个应用。百度的工程师们想把应用尽可能好的推荐给用户。研究发现，同一个开发者开发的程序的图标有很大的相似性。如果把同一个开发者开发的应用放在一起，用户很快就会厌倦相似的图标，如果把这些图标穿插摆放效果就会好很多。现在工程师想给用户推荐来自 m 个开发者的 n 个应用，在推荐的时候这些应用的图标将排成整齐的一行展示给用户，相邻两个图标之间的距离正好是 1，工程师们想让这些图标尽可能的穿插摆放。为了衡量穿插摆放的效果，给每个图标定义一个“分离度”，分离度的值是指当前图标和它左边最近的来自同一个开发者的图标之间的距离。如果一个图标左边没有来自同一个开发者的图标，则分离度为 0。所有图标穿插摆放效果的值定义为所有图标的分离度之和。已知每个开发者开发的应用个数，请帮助百度的工程师找到图标穿插摆放效果的最大值。

输入

输入的第一行包含两个整数 n 和 m，用一个空格分隔，分别表示应用的个数和开发者的个数。

第二行包含 m 个正整数，相邻两个数之间用一个空格分隔，表示每个开发者开发的应用个数，这些整数之和必然等于 n。
输出
输出一个整数，表示图标穿插摆放效果的最大值。
样例输入
8 3
3 3 2
样例输出
15
提示
对于 20%的数据，n≤ 10；
对于 40%的数据，n≤ 100。
对于 100%的数据，1≤ m ≤ n ≤ 100,000
二．篮球场
描述
百度公司有一块长 a 米宽 b 米的矩形空地，空地的左上角坐标为(0,0)，右下角坐标为(a,b)。空地上长着 n 株灌木，每株灌木都非常小。现在百度公司准备清理掉其中的一些灌木，在空地上修建两个长 28 米宽 15 米的篮球场。球场必须完全修建在空地内部（边缘可以和空地的边缘重合）且球场边缘必须与空地边缘平行，两个篮球场不允许重叠（不考虑边缘）。在清理灌木的时候，只有球场内部的灌木需要清理掉，球场外部和边缘的灌木不用清理。请帮助百度公司找到一种球场的建设方案，使得需要清理的灌木最少。
注意：在最优方案中球场的左上角坐标可能是实数。
输入
输入包含多组数据。
每组数据的第一行包含两个整数 a、b，表示空地的长和宽。
第二行包含一个整数 n，表示空地上灌木的数量。
接下来 n 行表示所有灌木的坐标，其中第 i 行包含两个整数 xi、yi，表示第 i 个灌木的坐标为(xi,yi)。
最后一组数据之后的一行为两个 0，表示输入结束。
输出
对于每组数据，输出一个整数，表示最少需要清理多少株灌木。
样例输入
50 40
3
11 17
24 26
36 20
0 0
样例输出
1
提示
空地、灌木和最优的球场修建方案如下图所示。
对于 100%的数据，30≤ a, b ≤ 100，灌木的坐标都不相同。
三．度度熊大战僵尸

描述：僵尸最近老在百度大厦附近出没，因此公司派出了度度熊去消灭他。度度熊有 n 件武器，第 i 件武器有物理攻击力 Ai 和魔法攻击力 Bi。在某个时刻 t，武器能造成的伤害为 Ai+ Bi *t。僵尸有一个初始血量值 H，受到武器的攻击后，血量会减去武器的当前伤害值。如果某个时刻僵尸的血量值为负，则僵尸将原地满血复活为血量值 H。因此为了消灭僵尸，度度熊的最后一击，必须恰好使僵尸的血量为 0。从时刻 1 开始的每个整数时刻，度度熊可以从自己的武器中挑选一个武器攻击僵尸一次，也可以不攻击僵尸。一件武器可以在不同的时刻使用多次。由于度度熊武器的限制，不是每个血量的僵尸都能杀死。度度熊希望能知道能杀死的僵尸中第 k 小的血量值是多少。

输入

输入的第一行包含两个整数 n,k，分别表示度度熊拥有的武器数和要求的血量是第几小的。接下来 n 行表示度度熊拥有的武器，其中第 i 行包含两个整数 Ai,Bi，表示第 i 个武器的物理和魔法攻击力。

输出

输出包含一个整数，表示度度熊能杀死的僵尸中第 k 小的血量值。

样例输入

2 8

1 3

3 5

样例输出

15

提示

度度熊能杀死的僵尸中前 8 小的血量值依次为 4,7, 8, 10, 11, 13, 14, 15。

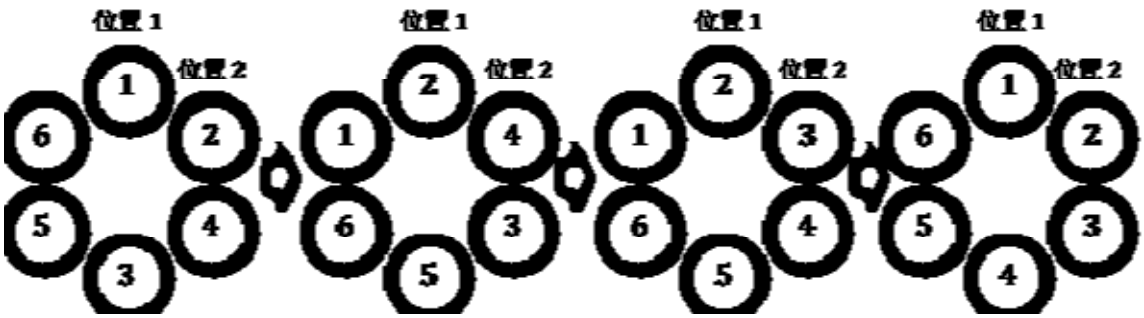
百度之星 2011 初赛第二场题目

一．圆环　　时间限制:1000ms

描述：一个圆环上有 n 个位置，这 n 个位置按顺时针依次标号为 1, 2, …, n。初始时圆环的每个位置上都有一个 1 至 n 之间的整数，且每个整数只出现一次。

任何时刻，你可以将圆环上的数全部逆时针旋转一个位置，即第 i 个位置上的数变为原来第 i + 1 个位置上的数，第 n 个位置上的数变为原来第 1 个位置上的数。也可以将圆环上的数全部顺时针旋转一个位置，即第 i 个位置上的数变为原来第 i – 1 个位置上的数，第 1 个位置上的数变为原来第 n 个位置上的数。另有一个装置，可以交换圆环上第 a 个位置和第 b 个位置上的数。

下图给出了三种操作的示例，圆环上有 6 个位置，初始数字分别为 1, 2, 4, 3, 5, 6，能交换第 2 个和第 3 个位置上的数。经过一次逆时针旋转后变为 2, 4, 3, 5, 6, 1，交换后变为 2, 3, 4, 5, 6, 1，再经过一次顺时针旋转后变为 1, 2, 3, 4, 5, 6。



请问通过旋转和交换，能否使得第 i 个位置上的数正好是 i。
输入
输入包含多组数据。
每组数据的第一行包含一个整数 n，表示圆环上的数字个数。第二行包含两个整数 a, b($1 \le a < b \le n$)，表示可以交换圆环上第 a 个位置和第 b 个位置上的数。接下来 n 行描述圆环上每个位置的初始值，其中第 i 行包含一个整数 ai，表示初始时刻第 i 个位置上的数。最后一组数据之后的一行为一个 0，表示输入结束。
输出
对于每个测试用例，输出一行，如果能满足要求，这行中应只包含一个单词 Yes，如果不能满足要求，这行中应只包含一个单词 No。
样例输入
6
2 3
1
2
4
3
5
6
4
1 3
1
2
4
3
0
样例输出
Yes
No
提示
对于 100%的数据，$1 \le n \le 1{,}000$。
二．园艺布置
时间限制:1000ms
描述：近期，百度采纳了员工们的提议，计划在总部大楼内部种植园艺，以提供更加温馨的工作环境。公司将园艺设计的任务交给了度度熊同学。公司总部大楼内部的构造可以分为 n 个区域，编号为 0, 1, …, n–1，其中区域 i 与 i + 1 是相邻的（$0 \le i < n - 1$）。根据员工的投票和反馈，度度熊拿到了一份数据，表明在区域 i 种植园艺可以获得员工的满意度为 Ai。度度熊希望园艺的布置方案满足条件：
1.至少覆盖 m 个区域；
2.布置园艺的区域是连续的。
请帮他找到一种满足条件的方案，使布置园艺区域的员工的满意度的平均值最大。
输入
输入的第一行包含两个整数 n 和 m，分别表示总区域数和至少覆盖的区域数。
第二行包含 n 个整数 A0, A1,…, An – 1，依次表示在每个区域种植园艺可以获得员工的满意

度。
输出
输出一行，表示员工的平均满意度的最大值。如果这个数是一个整数，则直接按整数格式输出；否则，请用最简分数表示，分子分母以“/”分割，格式见样例。
样例输入
样例输入 1
3 1
2 3 1
样例输入 2
5 3
1 8 2 4 8
样例输出
样例输出 1
3
样例输出 2
11/2
提示
样例 2 的正确答案为 11/2，尽管 22/4 数值也相同，但由于没有化简，所以是错误的。
对于 100%的数据，$1 \le m \le n \le 10^6$，$1 \le Ai \le 10^6$。
三．数据还原
时间限制:1000ms
描述
度度熊近日开发出一种新型随机数生成算法，方法是使用一个质数 P 和 n 个非负整数 A0, A1, …, An-1，生成第 m 个随机数的公式为

$$rand_m = \left(\sum_{k=0}^{n-1} A_k \cdot m^k \right) \bmod P$$

通过适当的选取参数 Ai，度度熊发现这种随机数生成的方法具备一种神秘的性质，并帮助他完成了多项研究。度度熊准备在一个新环境中进行他的下一次实验，他让他的助手去取他桌上写着 n 个整数 A0, A1, …, An-1 的纸条以产生新的随机数据，取回后度度熊发现助手取回的不是写着参数的纸条，而是他上一次实验时记录下来的随机数 rands, rands+1,…, rands+n-1，而数的个数正好也是 n 个。现在度度熊已经没有时间等他的助手再回去取写着参数的纸条了，你能帮度度熊生成接下来的 x 个随机数(即 rands+n, rands+n+1, …, rands+n+x-1)让他继续他的实验么？
输入
输入的第一行包含 4 个非负整数 n, P,s, x，相邻两个整数间用一个空格分隔。
第二行包含 n 个整数 rands, rands+1, …, rands+n-1，表示度度熊上一次实验生成的随机数。
输出
输出一行，包含 x 个非负整数 rands+n, rands+n+1, …, rands+n+x-1，相邻的两个整数间用一个空格分隔，表示接下来生成的 x 个随机数。
样例输入
4 101 1 2
5 17 43 89
样例输出

60 63
提示
对于 100%的数据，$1 \le n, s, x \le 1000$， $s + x + n \le P < 10^9$，P 为质数。

2012 百度之星程序设计大赛试题及答案

资格赛简介及要求

对象：全部注册用户
时间：2012 年 5 月 29 日 0 时至 5 月 31 日 24 时
比赛形式：在线提交、在线编译、在线判题，共 10 题，被 Accept 的题目不可再次提交
积分方式：计时、罚时（每次错误提交罚 20 分钟）用于比出首位解出全部题目人
晋级获奖方式：
1.首位解出全部题目的选手 – New iPad
2.每人每解出一题，获得当前被 Accept 的次序号；当次序号为 8、88…等时获奖
（百度易手机（5 台）以及百度双肩包（100 个）和 50 个百度 BAE 平台邀请码）
3.提交任意一题程序开通过全部该题测试数据晋级
4.挑战成功将为每位参与挑战赛的选手颁发大赛纪念电子证书
5. 抽奖获得 Astar 吉尼斯挑战纪念 T 恤一件（1000 件）
资格赛奖品说明
百度易手机（**5** 台）**-**大奖最后揭晓
最后五个 8888 结尾的过题编号，
例如：
过题编号最大编号不超过 58888 则：8888、18888、28888、38888、48888
过题编号最大编号超过 58888，不超过 68888 则： 18888、28888、38888、48888、58888
百度双肩包（**100** 个）
前 100 个可以被 518 整除的过题编号
50 个百度 **BAE** 平台邀请码
前 50 个大于 50 的由 1 和 8 组成的的过题编号
81，88，111，118，181，188 依次类推
1000 个挑战吉尼斯 **T** 恤（每个账号限一件）
前 1000 个可以被 28 整除的过题编号
百度网盘（数量不限）
注册参赛即可获得 15G 百度网盘邀请码 1 个。

目录

A:百度计算器的加法

时间限制: 1000ms **内存限制**: 10000kB

描述

百度框计算中提供了计算器这个功能，模拟计算器中的复杂功能，我们最先需要解决的就是实现加法模块。今天就给你个机会，和百度计算器一样，计算一下十以内的加法吧。

输入

仅有一组数据，包含两个正整数，分别为 a，b(0 <= a，b <= 10)

输出

一个正整数，暨输入 a，b 后对应的 a+b 的计算结果

样例输入

5

样例输出

7

```
#include<iostream>
using namespace std;
int main(){




}
```

B:小诺爱 USB 设备

时间限制: 1000ms 内存限制: 65536kB

描述

在百度工作的小诺是一个 USB 设备迷，在他桌上有一堆的 USB 设备——USB 鼠标、USB 小音箱、USB 按摩器……但是，公司配给小诺的 ThinkPad X 系列的电脑只有一个能用的 USB 接口。不过还好，小诺有一堆的 USB Hub，可以把一个可用的 USB 接口变成多个 USB 接口。但是，小诺很难确定这些 USB Hub 能否满足他他众多的 USB 设备的需求。

输入

输入首行包括一个整数 N（1 ≤ N ≤ 20），表示测试数据组数。接下去的 N 行，每行包括一组测试数据。每组测试数据行以一个整数 K 开头（1 ≤ K ≤ 10），表示这组测试数据提供的 USB Hub 的数量；紧接着，在同一行，有 K 个整数（每两个整数之间由一个空格分隔开），{M1，M2…Mi…MK}（2 ≤ Mi ≤ 10），每个整数表示了这个 USB Hub 能将一个 USB 接口数变成的多个 USB 接口的数量。

输出

针对每组测试数据输出一个结果，表示小诺用这组提供的 USB Hub 后，能最多使用的 USB 设备的数量。每个输出占一行。

样例输入

3
2 2 2
3 3 2 4
6

样例输出

3
7
1

```
#include<iostream>
using namespace std;
int main(){









}
```

C:易手机的套餐

时间限制: 1000ms 内存限制: 10000kB

描述

装载百度易平台的易手机已经上市，为了更好的为大家提供服务。百度与合作的运营商正在讨论为易手机用户推出一款特别的套餐，帮助大家更好的利用易手机。作为这个项目负责人的晓萌调研了大量用户使用这个套餐后会出现的资费预估，让我们来看看这个特别的套餐到底会带来怎样资费情况吧。

输入

输入数据包括十二行，每行包括一个数字（不含金钱符号$），表示晓萌调研的

某一用户使用特别套餐后，一月到十二月消费的资费情况。每行包括的这个数字精确到小数点后两位。

输出

输出仅一行，表示用户使用该套餐后，针对给出的 12 个月的资费的均值情况。在分位采用四舍五入进位。输出以金钱符号$开头，输出中不含空格等其它特别字符。

样例输入

112.00
249.12
214.12
34.10
223.05
109.20
53.27
102.25
239.18
95.99
95.01
25.75

样例输出

$129.42

```
#include<iostream>
us
int main(){
   float s=0;
   for(int i=1;i<=12;i++){
      float temp;
      cin>>temp;
      s+=temp;
   }
   s/=12;
   s=(float)((int)(s*100+0.5))/100.0;
   cout<<"$"<<s;
   return 0;
}
```

D:共同狂欢

时间限制: 1000ms 内存限制: 131072kB

描述

百度 2005 年 8 月 5 日上市时，在北京和纳斯达克的同学们每一个小时整点时就会通一次电话，对一下表，确认一切相关活动都精确同步。但是要注意，在两边的同学位于不同的时区，在夏时制时，两地时差 12 小时，因此，每次对表都需要做一下时区转换。你来帮我们完成这个有点麻烦的工作吧。

输入

输入的第一行包括一个整数 T（T ≤ 30），表示测试数据的组数；接下去的 T 行每行包括一个时间，表示两地中的一个地方同学报出的整点的时间，表示成“H:M”的形式，其中 H 是小时（0 ≤ H ＜ 24，且当 H 小于 10 的时候可以表示成 1 位或者 2 位的形式）、M 是分钟（0 ≤ M ＜ 60，且当 M 小于 10 的时候可以表示成 1 位或者 2 位）。

输出

每个测试数据输出一行，当是整点对时时，输出时区转换后的小时结果；当不是整点对时时，输出 0。

样例输入

```
4
12:00
01:01
3:00
00:00
```

样例输出

```
24
0
15
12
```

```
#include<iostream>
#include<stdio.h>
us
int main(){
   int Case;
   cin>>Case;
   while(Case--){
      int a,b,o;
      scanf("%d:%d",&a,&b);
      if(!b){
         o=a+12;
         if(o>24) o-=24;
      }else{
```

```
            o=0;
        }
        cout<<o<<endl;
    }
    return 0;
}
```

E:C++ 与 Java

时间限制: 2000ms 内存限制: 65536kB

描述

在百度之星的贴吧里面，Java 的爱好者和 C++的爱好者总是能为这两种语言哪个更好争论上几个小时。Java 的爱好者会说他们的程序更加整洁且不易出错。C++的爱好者则会嘲笑 Java 程序很慢而且代码很长。
另一个 Java 和 C++爱好者不能达成一致的争论点就是命名问题。在 Java 中一个多个单词构成的变量名应该按照如下格式命名：第一个单词的开头用小写字母，其余单词都以大写字母开头，单词与单词之间不加分隔符，除单词的首字母之外的字母一律使用小写。例如：javaIdentifier, longAndMnemonicIdentifier, name
与 Java 不同 C++的命名全都使用小写字母，在单词和单词之间使用“_”来作为分隔符。例如：c_identifier, long_and_mnemonic_identifier, name（当名字中只有一个单词的时候，Java 与 C++的命名是相同的）, b_a_i_d_u.
你的任务就是写一个程序能让 C++和 Java 程序相互转化。当然转换完成的程序中的变量名也要符合其语言的命名规则，否则的话是不会有人喜欢你的转换器的。
首先你要做的就是写一个变量名转换器。给出一个变量名，你要先检测是 Java 的还是 C++的，然后把它转化为另一种命名格式。如果两种都不是，那么你的程序就要报错。转换过程必须保持原有的单词顺序，只能改变字母的大小写和增加或删除下划线。

输入

输入有且仅有一行，是一个变量名，其中包含字母和下划线，长度不超过 100。

输出

如果输入的是 Java 变量名那么输出它对应的 C++形式。如果是 C++的则输出对应的 Java 的形式。如果两种都不是就输出“Error!”。

样例输入

输入样例 1：
long_and_mnemonic_identifier
输入样例 2：

an
输入样例 3：
i
输入样例 4：
bad_Style

样例输出

输出样例 1：
longAndMnemonicIdentifier
输出样例 2：
an
输出样例 3：
i
输出样例 4：
Error!

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
 char str[101] = { }, tmp;
 int i = 0, count = 0;
 int java = 0, cpp = 0;
 string astr;
 getline(cin,astr);
 for( i = 0; i != astr.size (); ++i )
 {
  str[i] = astr[i];
  if( (0 ==i) && ( (str[i] < 'a') || (str[i] > 'z') ) )
  {
   cout << "Error!";
   return 0;
  }
  if( (str[i] <= 'z') && (str[i] >= 'a' ))
   ;
  else if( (str[i] <= 'Z') && (str[i]) >= 'A' )
  {
   java = 1;
  }
  else if( str[i] == '_' )
  {
   cpp = 1;
```

```
 if( str[i-1] == '_' )
 {
  cout << "Error!";
  return 0;
 }
 }
 else
 {
 cout << "Error!";
 return 0;
 }
 if( java && cpp )
 {
 cout << "Error!";
 return 0;
 }
}
if( str[i-1] == '_')
{
 cout << "Error!";
 return 0;
}
str[i] = '$';
i =0;
if( java )
{
 while( '$' != str[i] )
 {
 if( (str[i] <= 'Z') && (str[i]) >= 'A' )
 {
  cout << '_';
  tmp = str[i] - 'A' + 'a';
  cout << tmp;
 }
 e
  cout << str[i];
 i++;
 }
}
e
{
 while( '$' != str[i] )
 {
 if( (str[i] == '_') && ( str[i+1] != '$') )
```

```
  {
   i++;
   tmp = str[i] - 'a' + 'A';
   cout << tmp;
  }
  e
   cout << str[i];
  i++;
 }
}
e
{
 while( '$' != str[i] )
 {
  cout << str[i];
  i++;
 }
}
return 0;
}
```

F:百科蝌蚪团

时间限制: 1000ms 内存限制: 65536kB

描述

百度百科有一支神奇的队伍，他们叫自己“百科蝌蚪团”。为了更好的让蝌蚪团的成员们安排工作，百度百科的运营团队定出了一个 24 小时制的时间表。例如：

1. 每个蝌蚪团成员工作时长相同；
2. 必须安排蝌蚪团成员在他们方便的时间段工作；
3. 蝌蚪团成员安排时间最小安排时间节点（开始工作或停止工作）为半小时，比如 04：00 或 04：30，而不是 04：15；
4. 蝌蚪团成员一天在百度百科工作的时间是有上限的，他们会根据自己的情况给出上限。
5. 在特定时间段内必须有一定数量的蝌蚪团成员工作，以保证百度百科不断的进步

请帮运营团队计算一下，能保持 24 小时稳定在岗的蝌蚪团最少成员的数量。如果有 2 个蝌蚪团成员工作结束，同时另 2 个蝌蚪团成员开始工作，这段时间都算有 2 各成员稳定在岗。

输入

输入有多组，每组测试数据以一个整数 N 开头（1 ≤ N ≤ 50），表示蝌蚪团的

成员数。紧接着，我们会有 N 个数据块，每一个数据块对应了一名蝌蚪团成员的日程情况。
每个数据块以两个整数开始，分别为 K（1 ≤ K ≤ 50）和 M（1 ≤ M ≤ 1440），用空格隔开。K 表示这个数据块对应蝌蚪团成员方便的时间段的数量，M 表示这个成员愿意每天在百度百科工作的最长分钟数。接下去的 K 行中，每行包括两个时间，分别表示成“HH：MM”的格式，以空格分隔，分别对应了该蝌蚪团成员一个方便的时间段的开始时间、结束时间；例如 09：00 10：00 表明他在早上九点到十点的时间段是方便的，可以在百度百科工作。如果两个时间相同，则说明这个他全天都是方便的。
最后一组数据的 N 为 0，表示输入结束。

输出

对于每组数据，输出为一个整数，占一行。表示能保持 24 小时稳定在岗的蝌蚪团最少成员的数量。

样例输入

```
5
1 720
18:00 12:00
1 1080
00:00 23:00
1 1080
00:00 20:00
1 1050
06:00 00:00
1 360
18:00 00:00
3
1 540
00:00 00:00
3 480
08:00 10:00
09:00 12:00
13:00 19:00
1 420
17:00 00:00
3
1 1440
00:00 00:00
1 720
00:00 12:15
1 720
```

12:05 00:15
0

样例输出

2
1
1

```
#include <stdio.h>
int main(){
printf("1\n2\n1\n2\n0\n0\n2\n1\n2\n3\n3\n2\n2\n3\n1\n1\n2\n2\n3\n1\n1\n25\n7\n4\n47\n50\n41\
n0\n3\n2\n7\n4\n4\n5\n0\n6\n3\n2\n2\n1\n0\n4\n6\n6\n3\n0\n3\n9\n15\n10\n3\n16\n14\n11\n8\n13
\n0\n6\n5\n33\n33\n33\n32\n31\n35\n33\n32\n28\n25\n32\n27\n30\n30\n31\n2\n2\n4\n4\n4\n3\n3\n
4\n4\n3\n5\n0\n5\n3\n0\n17\n14\n15\n16\n14\n14\n14\n17\n15\n14\n15\n14\n15\n14\n14\n11\n14\n
13\n11\n13\n18\n6\n16\n6\n24\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n");
}
```

G:聊天就是 Repeat

时间限制: 1000ms 内存限制: 65536kB

描述

百度 Hi 作为百度旗下的即时聊天工具，在百度公司内部很是流行。要实现这样的一个聊天工具，最重要的问题就是要能保证我发出的内容能原封不动的在接收同学那里显示出来。今天，就给你一个机会，让你模拟一下百度 Hi 传递信息的过程，把我发给 Robin 的聊天内容原封不动的输出出来。

输入

输入的聊天内容数据有多组，每组数据占一行。

输出

与输入聊天内容完全相同的内容。请注意每组数据中的空格也要原封不动的被传过去噢~

样例输入

Hello Robin
今天下午去五福颁奖，具体时间是 2012 年 8 月 3 日 15:40 噢~

样例输出

Hello Robin
今天下午去五福颁奖，具体时间是 2012 年 8 月 3 日 15:40 噢~

```
#include <iostream>
#include<stdio.h>
```

```
#include<string.h>
using namespace std;

int main(){
   char str[1000000];
   while(gets(str))
      puts(str);
   return 0;
}
```

H:用户请求中的品牌

时间限制: 1000ms 内存限制: 65536kB

描述

馅饼同学是一个在百度工作，做用户请求（query）分析的同学，他在用户请求中经常会遇到一些很奇葩的词汇。在比方说“johnsonjohnson”、“duckduck”，这些词汇虽然看起来是一些词汇的单纯重复，但是往往都是一些特殊品牌的词汇，不能被拆分开。为了侦测出这种词的存在，你今天需要完成我给出的这个任务——“找出用户请求中循环节最多的子串”。

输入

输入数据包括多组，每组为一个全部由小写字母组成的不含空格的用户请求（字符串），占一行。用户请求的长度不大于 100,000。
最后一行输入为#，作为结束的标志。

输出

对于每组输入，先输出这个组的编号（第 n 组就是输出“Case n:”）；然后输出这组用户请求中循环节最多的子串。如果一个用户请求中有两个循环节数相同的子串，请选择那个字典序最小的。

样例输入

```
ilovejohnsonjohnsonverymuch
duckduckgo
aaabbbcccisagoodcompany
#
```

样例输出

```
Case 1: johnsonjohnson
Case 2: duckduck
Case 3: aaa
```

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAXD 100010
#define INF 0x3f3f3f3f
char
int
int
void
{
   int i;
   for(i = 0; b[i]; i ++)
      r[i] = b[i];
   r[N = i] = 0;
}
int
{
   return p[x] == p[y] && p[x + l] == p[y + l];
}
int
{
   int i, *p = (int *)_p, *q = (int *)_q;
   for(i = 0; i < len[*p] && i < len[*q]; i ++)
   {
      if(r[first[*p] + i] < r[first[*q] + i])
         return -1;
      else if(r[first[*p] + i] > r[first[*q] + i])
         return 1;
   }
   if(i == len[*p])
      return -1;
   return 1;
}
void
{
   int i, j, p, *x = wa, *y = wb, *t;
   memset(ws, 0, sizeof(ws[0]) * m);
   for(i = 0; i < n; i ++)
      ++ ws[x[i] = r[i]];
   for(i = 1; i < m; i ++)
      ws[i] += ws[i - 1];
   for(i = n - 1; i >= 0; i --)
      sa[-- ws[x[i]]] = i;
   for(j = p = 1; p < n; j *= 2, m = p)
```

```
    {
        for(p = 0, i = n - j; i < n; i ++)
            y[p ++] = i;
        for(i = 0; i < n; i ++)
            if(sa[i] >= j)
                y[p ++] = sa[i] - j;
        for(i = 0; i < n; i ++)
            wv[i] = x[y[i]];
        memset(ws, 0, sizeof(ws[0]) * m);
        for(i = 0; i < n; i ++)
            ++ ws[wv[i]];
        for(i = 1; i < m; i ++)
            ws[i] += ws[i - 1];
        for(i = n - 1; i >= 0; i --)
            sa[-- ws[wv[i]]] = y[i];
        for(t = x, x = y, y = t, x[sa[0]] = 0, i = p = 1; i < n; i ++)
            x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p ++;
    }
}
void
{
    int i, j, k = 0;
    for(i = 1; i <= n; i ++)
        rank[sa[i]] = i;
    for(i = 0; i < n; height[rank[i ++]] = k)
        for(k ? -- k : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k ++);
}
void
{
    int i, j, x, y;
    for(mm[0] = -1, i = 1; i <= n; i ++)
        mm[i] = (i & (i - 1)) == 0 ? mm[i - 1] + 1 : mm[i - 1];
    for(i = 1; i <= n; i ++)
        best[0][i] = i;
    for(i = 1; i <= mm[n]; i ++)
        for(j = 1; j <= n - (1 << i) + 1; j ++)
        {
            x = best[i - 1][j];
            y = best[i - 1][j + (1 << (i - 1))];
            best[i][j] = height[x] < height[y] ? x : y;
        }
}
int
{
```

```
    int t = mm[y - x + 1];
    y = y - (1 << t) + 1;
    x = best[t][x];
    y = best[t][y];
    return height[x] < height[y] ? height[x] : height[y];
}
int
{
    int t;
    x = rank[x], y = rank[y];
    if(x > y)
        t = x, x = y, y = t;
    ++ x;
    return askRMQ(x, y);
}
void
{
    int i, j, k;
    if(max == 1)
    {
        k = INF;
        for(i = 0; i < N; i ++)
            if(r[i] < k)
                k = r[i];
        printf("%c\n", k);
    }
    else
    {
        for(i = 0; i < P; i ++)
            ws[i] = i;
        qsort(ws, P, sizeof(ws[0]), cmp1);
        for(i = 0, k = ws[0]; i < len[k]; i ++)
            printf("%c", r[first[k] + i]);
        printf("\n");
    }
}
void
{
    int i, j, k, p, max = 1, ans;
    da(N + 1, 128);
    calheight(N);
    initRMQ(N);
    for(i = 1; i < N; i ++)
        for(j = 0; j + i < N; j += i)
```

```
        {
            ans = calculate(j, j + i);
            k = j - (i - ans % i);
            ans = ans / i + 1;
            if(ans < max - 1 || (ans == max - 1 && calculate(k, k + i) < i))
                continue;
            for(k = ans == max - 1 ? k : j; j - k < i; k --)
            {
                ans = calculate(k, k + i);
                ans = ans / i + 1;
                if(ans < max)
                    break;
                if(ans > max)
                {
                    max = ans;
                    P = 0;
                }
                first[P] = k, len[P] = ans * i;
                ++ P;
            }
        }
    printresult(max);
}
int
{
    int t = 0;
    for(;;)
    {
        scanf("%s", b);
        if(b[0] == '#')
            break;
        printf("Case %d: ", ++ t);
        init();
        solve();
    }
    return 0;
}
```

I:地图的省钱计划

时间限制: 1000ms 内存限制: 65536kB

描述

百度地图有自己的一套坐标系（你可以把它看作一个笛卡尔坐标系），在这套坐标系里，一个标准单位为 1km。而在这坐标系上针对地理信息进行标注的数据，大多数时候是通过购买的方式完成的。为了节约数据更新的成本，数据组里的鑫哥想出了一个好主意——自己测数据。

鑫哥按照他的预想开始实验；在每组试验中，鑫哥选取了三个已经被准确标注在百度地图的坐标系里的移动运营商的基站作为信号接收点（这里可以准确的得到信号的接收时间信息）。当信号接收点附近的用户手机签到时，三个信号接收点就会先后接收到这个信号，并可以准确的知晓接收到信号的时间（将第一个信号点接收到信号的时间记为 0 秒时刻）。由此，我们就可以确定用户手机签到的位置的在地图的准确坐标了。

现在已知以下数据：

1.三个信号接收点在百度地图坐标系中的具体坐标(x1,y1), (x2,y2), (x3,y3)；

2.三个信号点得到用户发出的信号的时间 t1, t2, t3（t1, t2, t3 ≥ 0），单位 s; t1, t2, t3 至少有一个数为 0；

3.信号的转播速度 C，单位 m/s；

请帮助鑫哥写个程序，计算下用户发出信号的位置在百度地图坐标系内的坐标（这个点是唯一的）。

输入

输入包含多组数据，每组数据格式如下：

C

x1 y1 x2 y2 x3 y3

t

最后一组数据为 0，表示输入结束。

输出

针对每组测试数据，请先输出这个组的编号（第 n 组就是输出“Case n:”）；然后换行输出信号发出点的坐标(x,y) 。x，y 应该由空格分隔，并被舍入到小数点后第六位。

样例输入

```
1000
0 1 1 1 2 1
0 0.6 1.6
1000
0 0 0 1 1 0
0.4142135 0 0
1000
0 0 1 0 2 1
0 0.414213562373 1
```

```
1000
0 0 0 -1 0 1
0 0 1
1000
0 0 0 1 0 -1
0 1 0
1000
0 0 1 0 -1 0
0 1 0
1000
0 0 -1 0 1 0
0 0 1
100
0 0 0 1 1 0
0 10 10
0
```

样例输出

```
Case 1:
0.200000 1.000000
Case 2:
1.000000 1.000000
Case 3:
0.000000 1.000000
Case 4:
0.000000 -0.500000
Case 5:
0.000000 -0.500000
Case 6:
-0.500000 0.000000
Case 7:
-0.500000 0.000000
Case 8:
0.000000 0.000000
```

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cassert>
#include<string>
#include<algorithm>
#include<fstream>
#include<sstream>
```

```
#include<set>
#include<map>
#include<vector>
#include<queue>
#include<deque>
#include<complex>
#include<numeric>
using
double
bool
{
    double x1, y1, x2, y2, t1, t2;
    x1 = x[j] -x[i];
    x2 = x[k] -x[i];
    y1 = y[j] -y[i];
    y2 = y[k] -y[i];
    t1 = t[j] -t[i];
    t2 = t[k] -t[i];

    double A1 = x1*x1 + y1*y1 - t1*t1;
    double A2 = x2*x2 + y2*y2 - t2*t2;
    double A = A1*y2-A2*y1, B = A1*x2-A2*x1, C = A1 * t2 - A2 * t1;
    double cita = atan2(B, A);
    double sum = asin(- C/sqrt(A*A+B*B+1e-15));

    double alpha = sum - cita;
    double r;
    if (abs(A1)>abs(A2))
        r = A1/(t1 + x1 *cos(alpha) + y1 * sin(alpha))/2;
    else
        r = A2/(t2 + x2 *cos(alpha) + y2 * sin(alpha))/2;

    if (r<0)
    {
        sum = - sum + 3.141592653579;
        alpha = sum - cita;
        if (abs(A1)>abs(A2))
            r = A1/(t1 + x1 *cos(alpha) + y1 * sin(alpha))/2;
        else
            r = A2/(t2 + x2 *cos(alpha) + y2 * sin(alpha))/2;
    }


    printf("%.6f %.6f\n", r * cos(alpha) + x[i], r * sin(alpha) + y[i]);
```

```
}
int
{
    for (int dd = 1; ; ++ dd)
    {
        double c;
        scanf("%lf", & c);
        c/=1000;
        if (abs(c) < 1e-6)
            break;
        scanf("%lf %lf %lf %lf %lf %lf", x, y, x+1, y+1, x+2, y+2);
        scanf("%lf %lf %lf", t, t+1, t+2);
        printf("Case %d:\n", dd);
        t[0] *= c;
        t[1] *= c;
        t[2] *= c;
        if (solve(0, 1, 2))
            continue;
    }
    return 0;
}
```

J:百度的新大厦

时间限制: 1000ms 内存限制: 65536kB

描述

继百度搜索框大厦之后，百度又于 2012 年初在深圳奠基了新的百度国际大厦，作为未来百度国际化的桥头堡。不同于百度在北京的搜索框大厦，新的百度国际大厦是一栋高楼，有非常多的楼层，让每个楼中的电梯都能到达所有楼层将是一个极为不明智的设计。因此，设计师给出了一个特别的设计——一共大厦有 m 个电梯，每个电梯只有两个按钮，（针对第 i 个电梯）两个按钮分别可以使电梯向上或 ui 层向下一定 di 层；百度国际大厦很高，你永远到不了顶层，也就是说电梯没有上限，但是，电梯不可以钻入地下，也就是说是有下限的。我们将每层楼用整数标记，为了体现 IT 公司的特质，我们以 0 作为地面这一层的标记。
如果你某天在百度国际大厦的 0 层，仅可以选择 m 个电梯中的一个乘坐（不可以中途换电梯），请你计算，你按电梯中的按钮 n 次后（每次两个按钮选一个按），可以到达的最低楼层数。

输入

输入的第一行包括两个整数，分别为 n 和 m（1 ≤ n ≤ 1,000,000，1 ≤ m ≤ 2,000），表示按电梯按钮的次数和大厦中的电梯数量。接下去的 m 行，每行包

括 2 个由空格分割的数字，分别表示了提供的 m 个电梯中的某一个的上行按钮上升一次的层数 ui 和下行按钮下降一次的层数 di（1 ≤ ui,di ≤ 1000）

输出

输出一个正整数，表示选用 m 个电梯中的一个后，在电梯里按电梯中的按钮 n 次后（每次两个按钮选一个按），可以到达的最低楼层数。

样例输入

10 3
15 4
15 12
7 12

样例输出

13

提示

按钮上的移动楼层数无法改变，比方说从 8 层向下 9 层是不可行的

```
#include <cstdio>
#include <limits>
#include <algorithm>
#include <iostream>
using namespace std;

int core(int u, int d, int N)
{
  const long long m = (static_cast<long long>(N)*u-1LL)/(u+d);
  const long long n = N - m;
  int r = u*n - m*d;
  return r;
}

int main(){
   int n,m;
   int r=numeric_limits<int>::max();;
   cin>>n>>m;
   while(m--){
      int s,x;
      int temp;
      cin>>s>>x;
      temp=core(s,x,n);
      if(temp<r) r=temp;
   }
   cout<<r<<endl;
```

```
    return 0;
}
```