2009 北京冬令营

第一试

竞赛时间：2009 年 1 月 19 日 13:30 - 17:30

题目名称	难缠的值周生	函数依赖	电子竞技
可执行文件名	guard	dependence	compete
输入文件名	guard.in	dependence.in	compete.in
输出文件名	guard.out	dependence.out	compete.out
每个测试点时限	1 秒	1 秒	2 秒
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	无	无	无
题目类型	传统	传统	传统

提交源程序须加后缀

对于 Pascal 语言	guard.pas	dependence.pas	compete.pas
对于 C 语言	guard.c	dependence.c	compete.c
对于 C++ 语言	guard.cpp	dependence.cpp	compete.cpp

注意：最终测试时，所有编译命令均不打开任何优化开关

难缠的值周生

【问题描述】

小 P 上学总是迟到，迟到了以后常常会被值周生发现。被值周生发现就会给他所在的班级扣分，被扣了分不免要挨班主任的训，这令小 P 很不爽。不过，聪明的他经过观察发现，值周生通常会站在固定的位置，并且他们都很笨，只会向固定的同一个方向看。于是小 P 经过潜心研究，把值周生的站位和方向以及学校的地形绘成一张地图，你的任务是编写一个程序，帮助小 P 找出一条路从大门到达教室并且不被值周生发现。

【输入文件】

输入文件的第一行是两个整数 N，G，代表学校是一个 N × N 的正方形，其中有 G 个值周生。

接下来输入一个 N × N 的矩阵，代表学校。其中.代表空地，X(大写英文字母)代表墙，E 代表学校的大门，小 P 将从这里进入学校，C 代表教室，小 P 到达这里就胜利了。数字代表值周生，数字的值代表值周生的编号，显然值周生不能穿墙透视，但是小 P 也不能翻墙或者站到墙上去。

接下来有 G 行，每行包含一个数字 G_i 和一个字母，代表第 G_i 个值周生的朝向。其中 N 表示北，S 表示南，E 表示东，W 表示西。显然，如果你撞上值周生(走到值周生所在的格子上)，也会被值周生发现。

【输出文件】

输出文件包含任意一个行走方案，每行一个字母，表示小 P 每步走的方向。如果无论小 P 怎么走都不能不被值周生发现的走到教室，则输出一行整数-1。

【样例输入】

```
5 3
..E.1
...XX
.2.X.
.C.X.
...X3
1 E
2 E
3 W
```

【样例输出】

S
W
W
S
S
E
(注意这只是可能的行走方案之一)

【数据规模】

对于 100%的数据，满足 $0 \le G < 10$，$1 < N \le 10$。

函数依赖

【问题描述】

设 R(U)是一个属性集 U 上的关系模式，X 和 Y 是 U 的子集。若对于 R(U)的任意一个可能的关系 r，r 中不可能存在两个元组在 X 上的属性值相等，而在 Y 上的属性值不等， 则称 “X 函数确定 Y” 或 “Y 函数依赖于 X”，记作 X→Y。其中 X 称为这个函数依赖的决定属性集(Determinant)。

解释：如果有函数依赖 X→Y，当我们知道 X 的时候，也就知道了 Y，也就是 X 能推出 Y。另外，可以简单的证明，如果 X→Y，Y→Z，那么可以得到 X→Z。

若关系中的某一属性组的值能唯一的表示一个元组，则称该属性组为超码。

若关系中的某一属性组的值能唯一地表示一个元组，而其真子集不行，则称该属性组为候选码。

解释：设有属性集合(A, B, C)和函数依赖 A→B 和 B→C，显然，在已知 A 的情况下，我们能够通过函数依赖得到整个集合的所有属性 ABC，那么我们称 A 为超码。当超码的任何一个子集都不是超码的时候，我们称之为候选码。例如 AB 是一个超码，但是不是一个候选码，因为 A 是 AB 的子集，它也是超码。

现在给出属性集合和函数依赖集合 R(U, F)，请找出该 R 的候选码。

【输入文件】

输入文件的第一行为一个整数 N($N \le 10$)和 M($1 \le M \le 1000$)，表示属性集中的属性个数和函数依赖集中的依赖个数。这里我们默认大写字母中的前 N 个字母为我们所考虑的属性。接下来的 M 行每行一个字符串表示一个函数依赖，如 AB→DE。(中间的蕴含符号是由减号和大于号组成。另外需要说明的是，只有当我们同时得到 A 和 B 的时候，才能推出 D 和 E)。

【输出文件】

输出文件的第一行希望你输出你找到的候选码的个数 K。接下来的 K 行每行输出一个候选码。候选码本身按字母顺序排列，所有候选码按照字典顺序排列输出。如果没有找到候选码，输出“No candidate key”(不含引号)。

【样例输入】

```
5 5
AB->C
AC->B
AD->E
BC->D
E->A
```

【样例输入】

4
AB
AC
BE
CE

【数据规模】

对于30% 的测试数据，满足只有二元联系(即不存在函数依赖左边或右边的属性个数超过 1 个)。
对于 40%的测试数据，满足 $N \le 5$。
对于 70%的测试数据，满足 $M \le 100$。
对于100% 的测试数据，满足 $1 \le N \le 10, 1 \le M \le 1000$。

电子竞技

【问题描述】

每年的北京信息学冬令营上，同学们都会在课余开展丰富多彩的电子竞技活动。经过一年又一年的实战，同学们对彼此的实力已经非常熟悉了，以至于产生出了一套实力 – 对战评分系统。

这套系统把每个人的实力量化为一个整数。并且对于两个实力分别为 R_A, R_B 的人 A 和 B，这套系统把他们对战的通常的结果也量化为一个关于他们实力值的实数分数：

$$E_A = \frac{1}{1+10^{\frac{R_B-R_A}{400}}}$$

$$E_B = \frac{1}{1+10^{\frac{R_A-R_B}{400}}}$$

其中 E_A, E_B 分别为 A 和 B 在对战中通常得到的对战分数。例如，若 $R_A = 2432, R_B = 2611$，则在 A 和 B 的对战中 A 取得的对战分数应该为 $\frac{1}{1+10^{\frac{179}{400}}} \approx 0.263005239459$。

不过，今年冬令营的情况与以往略有不同，同学们分成了人数相等的两队展开了团体比赛。每队的老大给自己的队员排定一个顺序，然后两队按顺序第一个人互相对战，第二个人互相对战，以此类推。对于团体比赛中对战的每一对选手，我们都可以根据他们的实力值计算出他们分别获得的对战分数。而两个团队在这样一种比赛顺序下的团队对战分数就是团队每一位选手在这个顺序下获得的对战分数和。

为了公平竞赛，同学们约定了一个整数 L，并规定一个团队中实力为 x 的选手必须排在所有实力**严格小于** x – L 的选手之前。例如若 L = 100，实力为 2437 的一名选手必须排在本团队实力为 2336 的选手之前，但是却不一定排在本团队实力为 2337 的选手之前。

你作为一支队伍的老大，运用了种种不正当手段搞来了另外一直队伍排好的队员顺序。并且你也了解自己队伍和对方队伍每一个人的实力值。你现在的任务就是想方设法安排本队人员的顺序，使得你的团队在对战中获得的分数最大。

【输入文件】

输入文件的第一行包含一个整数 N，代表每队的人数。
接下来的 N 行，每行一个整数，表示你的每个队员的能力值。
接下来的 N 行，每行一个整数，**依次**表示对方团队的每个队员的能力值。
最后一行包含一个整数，代表 L。

【输出文件】

输出文件仅包含一个实数，表示你最大可能获得的分数。答案四舍五入保留到小数点后六位

【样例输入】

```
4
2239
2412
2399
2267
2534
2429
2340
2389
100
```

【样例输出】

```
1.483574
```

【样例说明】

根据题目中的规定，可能的安排顺序只可能有四种，它们能获得的分数如下

排列	分数
2399 2412 2239 2267	1.48040986
2399 2412 2267 2239	1.48357361
2412 2399 2239 2267	1.47815348
2412 2399 2267 2239	1.48131723

注意，如果没有 L 的限制，排列 2239 2267 2399 2412 可以得到更多的分数。但是，由于 2239 和 2267 不允许 2399 和 2412 之前出现，这是不可能的。

【数据规模】

对于100% 的数据，有 $1 \le N \le 20$ 成立。
对于100% 的数据，每个人的能力值都在 1500 到 3000 之间。
对于100% 的数据，有 $0 \le L \le 1500$ 成立。
对于100% 的数据，你的对手队伍排列好的顺序中的第 i 名队员的能力值与 L 的和大于等于对手队伍中排在他之后的任意一名队员的能力值。