1、打包行李

宰勋每次不到旅行前日都绝不会打包行李，今天也是到了登机的前一天才坐下来开始打包的。航空公司规定每人只能携带1件行李，而宰勋要带的东西比较多，1个行李箱肯定是不够的。下面的目录列出了他想要带的每件物品的体积和必要程度。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 物品 | 笔记本电脑 | 相机 | XBOX365 | 咖啡研磨机 | 哑铃 | 百科全书 |
| 体积 | 4 | 2 | 6 | 4 | 2 | 10 |
| 必要程度 | 7 | 10 | 6 | 7 | 5 | 4 |

因行李箱的空间有限，所以能够放进去的物品总体积不能超过w。编写程序计算出必要程度总和最大的物品目录。

示例输入值：

2

6 10

laptop 4 7

camera 2 10

xbox 6 6

grinder 4 7

dumbbell 2 5

encyclopedia 10 4

6 17

laptop 4 7

camera 2 10

xbox 6 6

grinder 4 7

dumbbell 2 5

encyclopedia 10 4

示例输出值：

24 3

laptop

camera

grinder

30 4

laptop

camera

xbox

grinder

2、计算第k个答案

摩尔斯电码字典

在没有电话的时代，摩尔斯电码是无线电传输领域中的一种常用代码。电码以短信号（短点，o）和长信号（长点，-）的不同组合表示各种文字。例如：o—表示英文字母J，而—表示英文字母M。

假设有一本以n个长点和m（n、m<=100）个短点组成的、包含所有信号的字典。例如：n=m=2，就会包含如下信号。

--oo

-o-o

-oo-

o--o

o-o-

oo--

这些信号已按照字典顺序排列好了。-的ASKII码是45，而o的ASCII码是111。因此，按照字典顺序，-在前，o在后。给定n和m时，编写代码计算出此字典的第k（k<=1,000,000,000,000）个信号。例如：上述字典的第四个信号是o--o。

3、剑客决斗

在路易十三和红衣主教黎塞留当权的时代，发生了一场决斗。n个人站成一个圈，依次抽签。抽中的人和他右边的人决斗，负者出圈。这场决斗的最终结果关键取决于决斗的顺序。现书籍任意两决斗中谁能胜出的信息，但“A赢了B”这种关系没有传递性。例如，A比B强，B比C强，C比A强。如果A和B先决斗，C最终会赢，但如果B和C决斗在先，则最后A会赢。显然，他们三人中的第一场决斗直接影响最终结果。

假设现在n个人围成一个圈，按顺序编上编号1~n。一共进行n-1场决斗。第一场，其中一人（设i号）和他右边的人（即i+1号，若i=n，其右边人则为1号）。负者被淘汰出圈外，由他旁边的人补上他的位置。已知n个人之间的强弱关系（即任意两个人之间输赢关系）。如果存在一种抽签方式使第k个人可能胜出，则我们说第k人有可能胜出，我们的任务是根据n个人的强弱关系，判断可能胜出的人数。

输入

第一行是一个整数N(1<=N<=20)表示测试数据的组数。

第二行是一个整数n表示决斗的总人数。(2<=n<=500)

随后的n行是一个n行n列的矩阵，矩阵中的第i行第j列如果为1表示第i个人与第j个人决斗时第i个人会胜出，为0则表示第i个人与第j个人决斗时第i个人会失败。

输出

对于每组测试数据，输出可能胜出的人数，每组输出占一行

样例输入

1

3

0 1 0

0 0 1

1 0 0

样例输出

3

4、第k个最大递增子序列

某个整数序列中，去掉0个以上的数字后，剩余的部分就是原序列的子序列。例如，{7,4,9}、{10,4}、{10,9}等是{10,7,4,9}的子序列。而序列{10, 4, 7}具有不同于原序列的排列顺序，因而不属于{10,7,4,9}的子序列。严格递增的子序列称为递增子序列。序列的递增子序列中，最长的序列称为最大递增子序列（LIS）。例如：{5,20,21,22,8,9,10}的最大递增子序列是{5,8,9,10}。（不唯一）

给出以不同数字组成（无重复数字）的序列时，请编写程序，计算此序列的LIS中按照字典序排在第k个位置的LIS。

输入

第一行输入测试用例的个数C（C<=50）。各测试用例的第一行输入序列中元素的个数n（1<=n<=500）和k（1<=k<=2\*109）。第二行输入序列的n个元素。各元素是大于等于1而小于等于100,000的整数，且同一数字只出现1次。

可以假设序列的LIS至少有k个。

输出

每个测试用例在第一行输出LIS的长度l，第二行以l个整数输出第k个LIS。

示例输入：

3

8 6

5 1 6 4 3 2 8 7

8 4

2 1 4 3 6 5 8 7

8 2

5 6 7 8 1 2 3 4

示例输出：

3

1 4 8

4

1 3 6 8

4

5 6 7 8

5、津巴布韦

由于计划经济失败，津巴布韦称为世界上通胀率最高的国家。这里的物价即使在一天中也会持续上涨，所以必须实时更新物品价格。例如：1个鸡蛋的价格为35亿津巴布韦元，所以超市做了每位数字的活动标价牌。

钟旭在穆加贝超市打工，有一天遇到了一位比较麻烦的客人。这位客人要退回刚才买走的鸡蛋，但是他不仅丢失了发票，而且连购买鸡蛋的数量也记不清了。鸡蛋价格已经在此期间上涨了1次，所以广告牌上已经写上新的价格。辛亏钟旭还记得如下两件事情。

1）最近一次价格上涨的时候，钟旭只是交换了塑料板的顺序。也就是说，没有添加其他塑料板，也没有去掉过广告牌中的塑料板。

2）看到最近一次上涨的价格时，钟旭心里曾经想过，“哇，这些钱刚好能购买m个糖果”。所以，最后的鸡蛋价格是m的倍数。（因为糖果的价格已经上涨，所以不能计算出鸡蛋的价格了）。

输入

第一行输入测试用例的个数C（C<=50）。之后的C行里面每行输入两个自然数e和m（1<=e<=1014，2<=m<=20）。当前鸡蛋的价格不能以0开始，但是之前的价格可以以0开始。

输出

每个测试用例在1行内输出可能的价格个数除以1 000 000 007的余数。

示例输入值：

4

321 3

123 3

422 2

12738173912 7

示例输出值

5

0

2

11033

示例输入输出值的说明

第一个示例输入值：以前鸡蛋的价格可能是123元、132元、213元、231元、312元。

第二个示例输入值：无论怎样重新排列123元的数字，结果都会比123元大，故无解。

第三个示例输入值：224元和242元是可能的价格。

第四个示例输入值：鸡蛋简直太贵了。

6、完全覆盖

有一天acmj在玩一种游戏----用2\*1或1\*2的骨牌把m\*n的棋盘完全覆盖。但他感觉把棋盘完全覆盖有点简单，他想能不能把完全覆盖的种数求出来？由于游戏难度增加他自己已经没法解决了，于是他希望大家能用程序来帮他把问题解决了。

输入

有多组数据。

每组数据占一行，有两个正整数n（0<n<12）,m(0<m<12)。

当n，m等于0时输入结束

输出

每组数据输出占一行，输出完全覆盖的种数。

样例输入

2 2

2 3

2 4

2 11

4 11

0 0

样例输出

2

3

5

144

51205

7、逃狱的汉尼拔博士

杀人狂魔汉尼拔博士逃狱了。通缉令发布后，大量军警出动并实施全天候追捕，不过狡猾的汉尼拔博士并没有落网。过了d日后，束手无策的警察们拜访了有着“编程天才”之称的查理教授。查理教授对汉尼拔博士留在监狱的笔记本进行分析后，做出了如下假设。

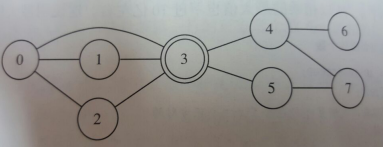
1）汉尼拔博士为了避开检查，只走山路；

2）汉尼拔博士越狱当天选择了与监狱相邻的村子之一作为藏身之处；

3）汉尼拔博士为了逃避追捕，每天往一个相邻的村子逃窜。

为了验证假设，教授找到了与监狱所在村子以山路连接的n个村子的地图。汉尼拔博士会按照此假设行动，而且会随机选择一个备选的村子。编写程序计算d日后汉尼拔博士在各个村子的概率。

例如监狱在第三个村子，逃狱后的汉尼拔博士会在0、1、2、4、5中任意选择一个村子藏身。因此，1天后汉尼拔博士藏在第0号村子的概率是1/5，两天后藏在第1号村子的概率是1/15。



输入

第一行输入测试用例的个数C（1≤C≤50）。之后各行输入地图上显示的村子个数N（2≤N≤50）和逃狱后经过的天数D（1≤D≤100），以及监狱所在村子的号码P（0≤P<N），村子的号码由0到N-1的数字组成。之后N行里各输入N个整数，形成一个序列A。第i行j列的数值A[i][j]如果等于1，就表示从第i号村子到第j号村子有山路可走；如果是0，则表示无路可通。接下来的一行输入要计算概率的村子的个数T（0≤T<N），最后一行以整数型输入要计算概率的村子的号码Q（0≤Q<N）。

如果一个村子与另一个村子相连，那么相反的路径也必定存在。可假设一个村子连接到自身的路径不存在。

输出

每个测试用例以T个实数输出汉尼拔博士可能藏匿的概率。存在小于10-7的绝对/相对误差的答案将被视为正确答案。

示例输入值

2

5 2 0

0 1 1 1 0

1 0 0 0 1

1 0 0 0 0

1 0 0 0 0

0 1 0 0 0

3

0 2 4

8 2 3

0 1 1 1 0 0 0 0

1 0 0 1 0 0 0 0

1 0 0 1 0 0 0 0

1 1 1 0 1 1 0 0

0 0 0 1 0 0 1 1

0 0 0 1 0 0 0 1

0 0 0 0 1 0 0 0

0 0 0 0 1 1 0 0

4

3 1 2 6

示例输出值

0.83333333 0.00000000 0.16666667

0.43333333 0.06666667 0.06666667 0.06666667

8、回转寿司

algospot内部举行的解题投注比赛中，积累的罚金实在太多，于是运营团队决定举行一次会餐，地点选择在回转寿司店。来到回转寿司店的运营团队并没有急于品尝寿司，而是开了一个“策略”会议。寿司店里共有n种菜品，团队对各个菜品标上了如下等级。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 寿司种类 | 鸡蛋 | 三文鱼 | 鳗鱼 | 金枪鱼 | 牛排 | 炸鸡 |
| 价格 | 2500 | 3000 | 4000 | 5000 | 10000 | 15000 |
| 等级 | 7 | 9 | 10 | 12 | 20 | 1 |

运营团队要在不超过预算的情况下吃到等级总和最大的菜品。假设可供购买的寿司不限量，那么能吃到的最大等级之和是多少呢？

输入

第一行输入测试用例的个数C（1<=C<=5）。各测试用例的第一行输入寿司的种类n（1<=n<=20）和团队的总预算m（1<=m<=109）。之后的n行中，每行输入一种寿司的价格和等级。价格是2000以下的整数，但必须是100的倍数。等级是20以下的自然数。

输出

每个测试用例在1行内输出可能的最大等级之和。

示例输入值

2

6 10000

2500 7

3000 9

4000 10

5000 12

10000 20

15000 1

6 543975612

2500 7

3000 9

4000 10

5000 12

10000 20

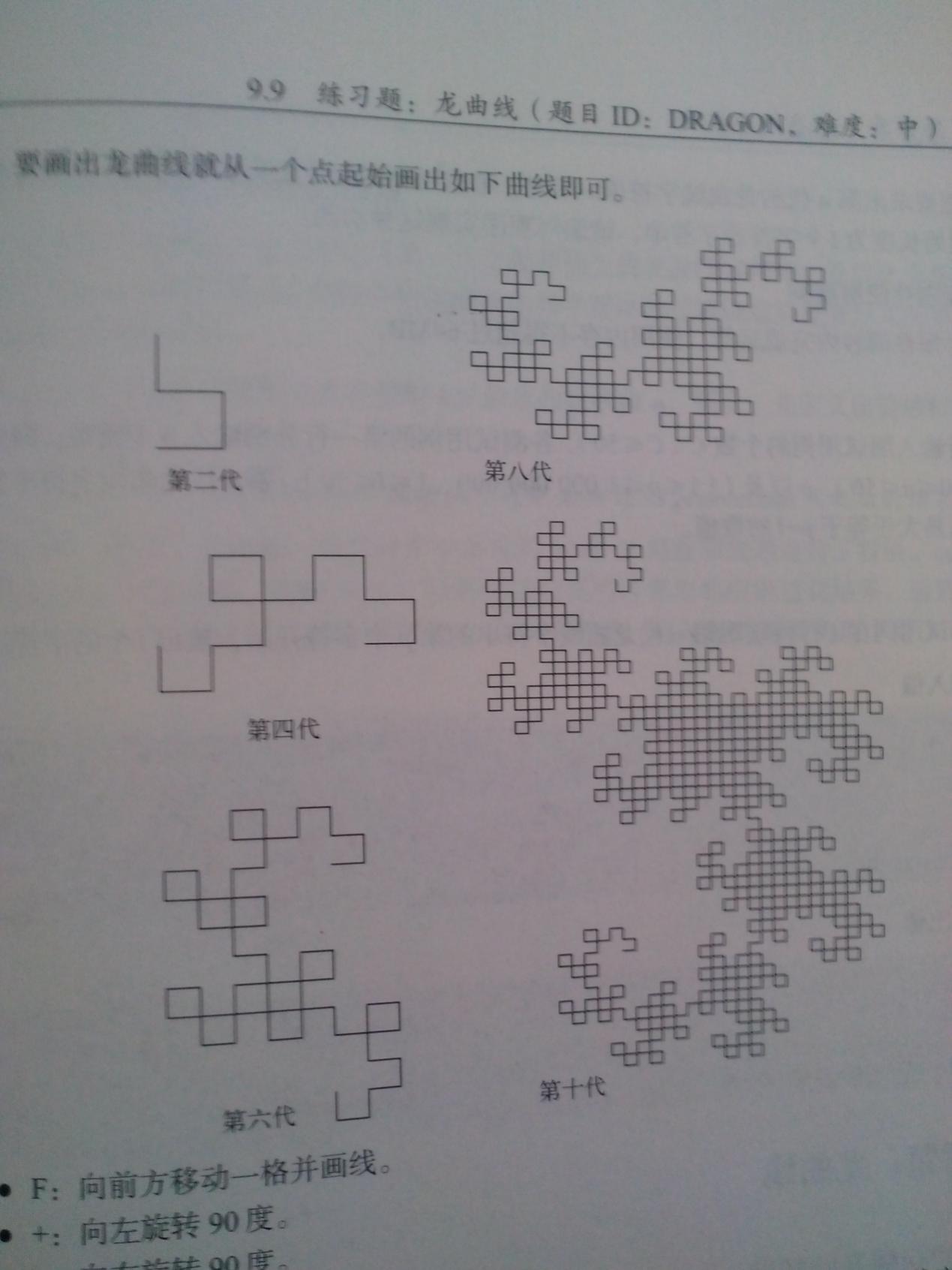
15000 1

9、龙曲线

龙曲线是以简单的数学规则画出一种曲线，它具有以下形态。曲线从一个简单的线段起始，按照一定规则变换此线段完成整个曲线。每形成一次变换称为“完成了一次变换代”，而每完成一代，曲线会进化到更复杂的形式。像这种“放大其一小部分的形状时，表现出与整个形状极为相似构造的图形”，就是分形。

画出龙曲线的方法暂且就称为龙曲线字符串吧！龙曲线字符串由X、Y、F、+、-组成。

那么，要画出龙曲线就从一个点起始画出如下曲线即可。



F：向前方移动一格并画线。

+：向左旋转90度。

-：向右旋转90度。

X、Y：忽略。

画出第0代龙曲线的字符串是FX。从下一代开始，按照如下方式利用前一代字符串进行字符替换，从而获得当前一代的龙曲线字符串。

X-> X+YF

Y-> FX+Y

根据上面的替换式，就有如下的1、2代龙曲线字符串。

第一代：FX+YF

第二代：FX+YF+FX-YF

我们想要求出第n代龙曲线字符串。不过，考虑到答案有可能很长，所以只想计算出第p个字符起始长度为l个字符的字符串。请编写程序实现这种功能。

输入

第一行输入测试用例的个数C（C<=50）。各测试用例的第一行分别输入3个整数，即龙曲线的世代n（0<=n<=50）、p以及l（1<=p<=1 000 000 000、1<=l<=50）。第n代龙曲线字符串的长度可假设成总是大于等于p+l的数值。

输出

每个测试用例在1行内输出第n代龙曲线字符串的第p个字符开始，输出l个字符。

示例输入

4

0 1 2

1 1 5

2 6 5

42 764853475 30

示例输出

FX

FY+YF

+FX-Y

FX-YF-FX+YF+FX-YF-FX+YF-FX-YF-

10、通配符

通配符在很多操作系统中只用部分文件名指定文件。这些加有通配符的字符串就是通配符范式，这种范式与文件名类似，但常常是包含特殊字符“\*”或“？”的字符串。

从通配符范式的第一个字符开始与文件名比较，如果所有字符都一致，那么通配符范式与文件名相对应。通配符范式中的“？”字符可以充当任何一个字符，而“\*”字符可以充当长度大于等于0的任一字符串。例如，通配符范式he?p可表示help、heap，但不能表示helpp。而通配符范式“\*p\*”可表示help、papa，但不能表示hello。

下面给定通配符范式和文件名集合，编写程序找出对应于通配符范式的文件名。

输入

第一行输入测试用例的个数C（1<=C<=10）。各测试用例的第一行输入通配符范式W，第二行输入文件名数量n（1<=n<=50）。接下来的n行中，每行输入1个文件名。通配符范式由大小写英文字母、数字、\*、？组成，文件名由大小写英文字母和数字组成。所有字符串的长度都大于等于1小于等于100，且不包含空格。

输出

每个测试用例将按照字母顺序每行显示1个对应于通配符范式的文件名。

示例输入

3

he?p

3

help

heap

helpp

\*p\*

3

help

papa

hello

\*bb\*

1

babbbc

示例输出

heap

help

help

papa

babbbc

设计通配符匹配算法，其中\*号可以匹配任意多个字符，？号可以匹配任意一个字符。例如12345和12\*、12\*？以及12\*4？等都匹配。

函数原型为：bool match(const char\* str, const char\* strpattern);

分析：利用动态规划来解决。

此题类似与LCS问题。假设字符串A[i]代表前i+1个子字符串，B[j]代表前j+1个子字符串，那么A[i]与B[j]是否匹配可以由A[i-1],B[j-1];A[[i-1],B[j];以及A[i]B[j-1]结合当前A[i]与B[j]的字符来确定。

A[i]与B[j]匹配需进行如下判断

1、若(strpattern[j-1] == '?' && str[i-1] != '\0')或者strpattern[j-1]==str[i-1]，即A[i]与B[j]的最后一个字符必须“相同”(这里相同是指必须占用一个字符)，则只需判断A[i-1]和B[j-1]是否匹配；

2、若strpattern[j-1] == '\*'，则只需A[i]和B[j-1]匹配或者A[i-1]和B[j-1]匹配或者A[i-1]和B[j]匹配即可确定A[i]与B[j]匹配.

11、量化

量化过程将宽范围的数值表示为小范围的近似值，从而达到有损压缩的目的。例如：16位的JPG文件转换成4色的GIF文件，就是把RGB颜色空间的颜色量化成4种颜色的过程。还有，把身高为161、164、178、184的4名学生表示成“160-169阶段的2名，170-179阶段的1名、180以上阶段的1名”的方式也是量化。

现要把小于1000的自然数组成的序列量化成s个自然数组成的序列。量化的方法其实很多，例如，只用两个数表示序列{1 2 3 4 5 6 7 8 9 10}，就可以表示成{3 3 3 3 3 7 7 7 7 7}。或还可以用{1 1 1 1 1 10 10 10 10 10}的形式表示。那么，各数值误差平方之和最小的量化结果是多少呢？

比如，把序列{1 2 3 4 5}量化成{2 2 3 3 3}，各数值量化后的误差是-1、0、0、1、2。那么，误差平方之和是1+0+0+1+4=6。但是，如果量化成{2 2 3 4 4}，那么误差平方之和是1+0+1+0+1=3。接下来请编写程序，求用s个数值量化给定序列后，误差平方之和的最小值。

输入

第一行输入测试用例的个数C（1<=C<=50）。各测试用例的第一行输入序列的长度n（1<=n<=100），以及可使用的数值个数s（1<=s<=10）。第二行输入序列的n个整数，且序列的所有数值是1000以下的自然数。

输出

每个测试用例将在1行内输出用s个数值量化给定序列时产生的最小误差平方之和。

示例输入值

2

10 3

3 3 3 1 2 3 2 2 2 1

9 3

1 744 755 4 897 902 890 6 777

示例输出值

0

651

12、探寻宝藏

传说HMH大沙漠中有一个M\*N迷宫，里面藏有许多宝物。某天，Dr.Kong找到了迷宫的地图，他发现迷宫内处处有宝物，最珍贵的宝物就藏在右下角，迷宫的进出口在左上角。当然，迷宫中的通路不是平坦的，到处都是陷阱。Dr.Kong决定让他的机器人卡多去探险。

但机器人卡多从左上角走到右下角时，只会向下走或者向右走。从右下角往回走到左上角时，只会向上走或者向左走，而且卡多不走回头路。（即：一个点最多经过一次）。当然卡多顺手也拿走沿路的每个宝物。

Dr.Kong希望他的机器人卡多尽量多地带出宝物。请你编写程序，帮助Dr.Kong计算一下，卡多最多能带出多少宝物。

输入

第一行： K 表示有多少组测试数据。

接下来对每组测试数据：

第1行: M N

第2~M+1行： Ai1 Ai2 ……AiN (i=1,…..,m)

【约束条件】

2≤k≤5 1≤M, N≤50 0≤Aij≤100 (i=1,….,M; j=1,…,N)

所有数据都是整数。 数据之间有一个空格。

输出

对于每组测试数据，输出一行：机器人卡多携带出最多价值的宝物数

样例输入

2

2 3

0 10 10

10 10 80

3 3

0 3 9

2 8 5

5 7 100

样例输出

120

134

这是双进程DP问题，首先，假设出发点为A 终点为B 那么，根据题目给出的条件，可以推出A->B的动态转移方程为 dp[i][j] = max(dp[i-1][j],dp[i][j-1]) + a[i][j]; 由于，同理可得B的情况，那么，题目的意思是A->B 然后 B -> A我们可以假设同时从A点出发，得到两条不同路径，这个是一样的效果。所以，我们可以得到一个动态转移方程  
dp[i][j][p][q] = max(dp[i-1][j][p-1][q],dp[i-1][j][p][q],dp[i][j-1][p-1][q],dp[i][j-1][p][q-1]) 因为 每次只能移动一步，即 i+1 或j+1 那么 i+j是移动的步数 因为从A点开始移动的，经过相同的步数，肯定能得到i+j = p+q

还有一点要注意一下，这题与NYOJ 61是同类问题，但是，有一点细节要注意，最后终点的值也要算上，上面的动态方程得到的值不包含两个A 和 B的值，因为 A是起点，所以，他的值一般是0，所以，得到最后的结果应该是 int sum = max(dp[m-1][n][m-1][n],dp[m-1][n][m][n-1],dp[m][n-1][m-1][n],dp[m][n-1][m][n-1]) + a[m][n];