ACM

◎ 赵端阳 袁 鹤 编著

国际大学生程序设计竞赛题解（1）

http: //www.phei.com.cn

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

ACM 国际大学生程序设计竞赛题解（1）

赵端阳　袁　鹤　编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

随着各大专院校参加 ACM/ICPC 热情的高涨，迫切需要有关介绍 ACM 国际大学生程序设计竞赛题解的书籍。本书根据浙江大学在线题库的部分题目，经过筛选、分类后进行了解答（个别特别简单或者特别复杂的题目未选择），比较详细地分析和深入浅出地讲解了解题的方法和用到的算法。题目的类型包括基础编程、模拟、字符串处理、搜索、动态规划、贪心、图论、几何和数学题。

本书可以作为高等院校有关专业的本科和大专学生参加国际大学生程序设计竞赛的辅导教材，或者作为高等院校数据结构、C/C++程序设计或算法设计与分析等相关课程的教学参考书。

书中各题目的源代码及相关资料可在 http://www.hxedu.com.cn 下载。

图书在版编目（CIP）数据

ACM 国际大学生程序设计竞赛题解. 1 / 赵端阳，袁鹤编著. —北京：电子工业出版社, 2010.6
ISBN 978-7-121-11063-4

Ⅰ. ①A… Ⅱ. ①赵… ②袁… Ⅲ. ①程序设计－竞赛－高等学校－解题 Ⅳ. ①TP311.1-44

中国版本图书馆 CIP 数据核字（2010）第 107789 号

责任编辑：陈晓莉
印　　刷：北京天宇星印刷厂
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：23.75　字数：608 千字
印　　次：2010 年 6 月第 1 次印刷
印　　数：4000 册　　定价：38.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

ACM 国际大学生程序设计竞赛（ACM/ICPC：ACM International Collegiate Programming Contest）是由国际计算机界历史悠久、颇具权威性的组织 ACM 学会（Association for Computing Machinery，美国计算机协会）主办，是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛，其目的旨在使大学生运用计算机来充分展示自己分析问题和解决问题的能力。1970 年在美国 Texas a&m 大学举办了首次区域竞赛，从而拉开了国际大学生程序设计竞赛的序幕。该项竞赛从 1970 年举办至今已经历 29 届，因历届竞赛都荟萃了世界各大洲的精英，云集了计算机界的“希望之星”，而受到国际各知名大学的重视，并受到全世界各著名计算机公司的高度关注，成为世界各国大学生最具影响力的国际级计算机类的赛事。

此项赛事的举办目的不单是培养参赛选手的创造力、团队合作精神以及他们在软件程序开发过程中的创新意识，同时也是检测选手们在压力下进行开发活动的能力。因此，ACM 国际大学生程序设计竞赛是参赛选手展示计算机才华的广阔舞台，是著名大学计算机教育成果的直接体现，是信息企业与世界顶尖计算机人才对话的最好机会。该项竞赛分区域预赛和国际决赛两个阶段进行，各预赛区第一名自动获得参加世界决赛的资格，世界决赛安排在每年的 3～4 月举行，而区域预赛安排在上一年的 9～12 月在各大洲举行。从 1998 年开始，IBM 公司连续赞助该项赛事的世界决赛和区域预赛。这项比赛是以大学为单位组队（每支队伍由教练、3 名正式队员，一名后备队员组成）参赛的。

国际 ACM 比赛 1996 年才进入中国内地，上海交通大学作为我国内地高校最早的参赛队之一，曾 7 次进军总决赛，并于 2002 年将 ACM 金杯首次带到亚洲。打破了几十年来欧美国家对这一赛事绝对的统治地位，更震惊了世界。

2005 年 4 月 6 日，在上海举办的“第 29 届 ACM 国际大学生程序设计竞赛”总决赛中，上海交通大学团队成功解答 8 道题，以一题领先的优势力克来自世界六大洲 29 个国家和地区的 78 支参赛队，捧获“世界上最聪明的人”的冠军奖杯。时隔三年，ACM 全球总决赛冠军奖杯再次回到了上海交通大学，领队俞勇教授与队员们手举国旗和校旗走上最高领奖台，激动之泪夺眶而出。上海交通大学团队从招生选苗、选拔队员、日常训练、寒暑假集训、赛前封闭训练，日复一日，年复一年，凭着顽强的毅力、强大的凝聚力，一步一个脚印、一步一个台阶，不断积累，顽强拼搏，两次携手捧回世界杯。

国际 ACM 比赛是世界上规模最大、历史最长、影响最深的全球性计算机专业竞赛，它要求每一名队员不仅具有扎实的数学功底，非凡的算法设计能力，娴熟的编程技巧，而且必须具备很好的协作精神、稳定的心理素质、快速的临场应变能力。

为了帮助各个大专院校的大学生们了解国际大学生程序设计竞赛，了解其程序设计的方法，提高参与校级、省级和亚洲区赛国际大学生程序设计竞赛的兴趣，特编写这本题解。

全书共分 9 章：第一章，基础编程题。主要是比较容易的题目，也称简单题，尤其适合刚刚开始熟悉 ACM 大学生程序设计竞赛做题的同学。通过这些题目的练习，主要是熟悉数据的输入、输出格式，基本编程方法，在线提交系统的使用，常见错误及其对策。第二章，模拟算

法题。根据题目的要求逐步实现目标，虽然没有经典的算法可以使用，正确理解题目是关键的要素。例如题目 Enigma、Enigma2 和 Microprocessor Simulation 等。第三章，字符串处理题。主要是字符串的处理，这也是考察参赛者细心的一类题目，需要对各种情况进行仔细的分析，予以正确的处理。例如题目 Operand 和 P,MTHBGWB 等。第四章，基本数据结构题。常见的有二叉树、堆栈和列表等，如 Trees Made to Order 等。第五章，搜索算法题。这是一个最常见的类型，通常有深度优先搜索和广度优先搜索算法。例如题目 Fire Net 和 Robotic Jigsaw 等。第六章，动态规划算法题。这些题目使用动态规划算法，会获得较快的运行时间和效率，例如题目 Great Equipment 等。第七章，贪心算法题。这些题目使用贪心算法，会获得较快的运行时间和效率，例如题目 Mainframe 和 Gene Assembly 等。第八章，图论算法题。主要包含拓扑排序、Floyd 算法和二分图等，例如题目 FDNY to the Rescue!和 Roads Scholar 等。第九章，几何和数学题。主要是几何题和一些数学计算题目，例如题目 Farmland 和 Transmitters 等。

本书通过大量的实例介绍了竞赛中常用的算法，并对如何灵活地应用这些算法进行了比较详细的分析和深入浅出的讲解。

本书所用的语言是 C/C++，并在 MinGW Developer Studio 中调试通过。在运行时间和内存占用的性能方面，C++并不比 C 语言优越多少，只是 C++有丰富的模板库函数，输入/输出语句简单，编写的代码看起来格式更工整而已。在有大量输入/输出数据的题目中，书中会特别提醒读者需要使用 C 语言的输入/输出。

显然，“Accepted”是我们的目标，算法是我们不断探索的道路，好的算法让我们容易达到目标。本书中的算法，虽然作者经过反复斟酌，不断优化和简化，仍然不能认为是最优的。因为对算法的探索，正如金庸笔下的大侠们苦练武功一样，是没有止境的。

本书中虽然描述的是“ACM 国际大学生程序设计竞赛”中最基本的算法，但它也已经超出了一般本科教科书所讲授的范围。清华大学计算机科学与技术系博士生导师、国际信息学奥林匹克中国队总教练吴文虎教授认为算法的确是艺术，“艺术与科学是相通的，都会给人以美的享受。”并感叹道：“体味思维艺术之美，我以为这可能是更高层次的享受”[1]。希望读者能从本书中体会到这一点。

本书可以作为高等院校有关专业的本科和大专学生参加国际大学生程序设计竞赛的辅导教材，或者作为高等院校数据结构、C/C++程序设计或算法设计与分析等相关课程的教学参考书，旨在培养和提高学生参加 ACM 国际大学生程序设计竞赛的兴趣。

本书在编写过程中得到了浙江大学在线裁判系统管理员的大力支持，在此表示非常感谢。

感谢沈仙桥、郑丹同学翻译了大部分的题目。

同时也得到了 ACM 参赛队员戚勇、林刚、王海江、沈懿华、郑彦良、戴俊、周智栋、应建伟等同学的热情帮助。

由于作者水平所限，书中难免有不足之处，恳请广大读者批评指正。编著者电子邮件地址：727946579@qq.com 和 hzyuanhe@163.com。

编著者

2010 年 4 月于杭州

[1] 吴文虎主编，孙贺编著，序，程序设计中的组合数学，清华大学出版社，2005 年 5 月。

目　录

第一章　基础编程题

在本章的题目大部分都比较简单，作为刚刚开始参加竞赛的练习题。

ZJU1037-Gridland[1、2、3]

Time limit: 1 Seconds　　Memory limit: 32768K

Background

For years, computer scientists have been trying to find efficient solutions to different computing problems. For some of them efficient algorithms are already available, these are the "easy" problems like sorting, evaluating a polynomial or finding the shortest path in a graph. For the "hard" ones only exponential-time algorithms are known. The traveling-salesman problem belongs to this latter group. Given a set of N towns and roads between these towns, the problem is to compute the shortest path allowing a salesman to visit each of the towns once and only once and return to the starting point.

Problem

The president of Gridland has hired you to design a program that calculates the length of the shortest traveling-salesman tour for the towns in the country. In Gridland, there is one town at each of the points of a rectangular grid. Roads run from every town in the directions North, Northwest, West, Southwest, South, Southeast, East, and Northeast, provided that there is a neighbouring town in that direction. The distance between neighbouring towns in directions North-South or East-West is 1 unit. The length of the roads is measured by the Euclidean distance. For example, the figure below shows 2×3-Gridland, i.e., a rectangular grid of dimensions 2 by 3. In 2 × 3-Gridland, the shortest tour has length 6.

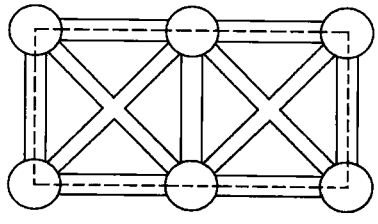


Figure 1-1　A traveling-salesman tour in 2×3-Gridland.

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1037

[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=1450

[3] http://acm.uva.es/archive/nuevoportal/data/problem.php?p=2334

Input

The first line contains the number of scenarios.

For each scenario, the grid dimensions m and n will be given as two integer numbers in a single line, separated by a single blank, satisfying $1<m<50$ and $1<n<50$.

Output

The output for each scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. In the next line, print the length of the shortest traveling-salesman tour rounded to two decimal digits. The output for every scenario ends with a blank line.

Sample Input

2
2 2
2 3

Sample Output

Scenario #1:
4.00

Scenario #2:
6.00

Problem Source

Northwestern Europe 2001

【题目大意】

背景：

多年来，计算机科学家一直在寻找有效的方法来解决困难的计算问题。有些问题已经找到了有效的算法，如排序、计算多边形面积、寻找图的最短路径，这些都是“容易的”问题。而“困难的”问题，目前只有计算时间是指数级的算法，旅行售货员问题就是其中之一。给出 N 个城市，及城市之间的道路，问题是寻找一条最短的路径，让售货员访问每个城市一次且只有一次，又回到出发点。

问题：

Gridland 的总统雇用你来编写一个程序，计算在这个国家中所有城市的旅行售货员问题的最短长度。在 Gridland，每个城市都位于矩形网格的一个点上。通向每个城市的道路只有东、西、南、北、东南、东北、西南和西北方向，在每个方向上只有一个相邻的城市。东西、南北方向上的长度为 1，长度单位是欧几里得距离。如图 1-1 所示是一个 2×3 的 Gridland，最短的环游路线长度是 6。

输入：

第一行是测试例数。

每个测试例，在一行里有两个整数 m 和 n（$1<m, n<50$），中间有一个空格，是二维栅格的大小。

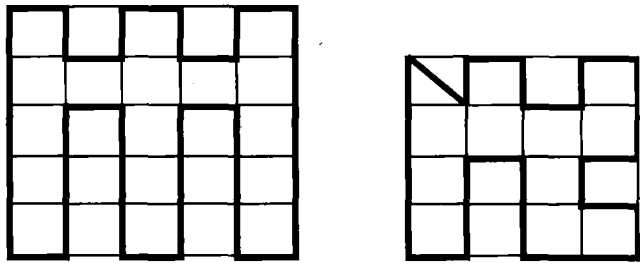
输出：

对每个测试例，第一行输出："Scenario #i:"，i 是从 1 开始的测试例编号。第二行输出售货员环游问题的最短路长度，精确到两位小数。每个测试例之后有一个空行。

【算法分析】

题目中一直强调旅行售货员问题，容易诱导竞赛者去使用复杂的算法，如回溯算法、分枝限界算法。而本题的城市位置是特殊的，边长也是特殊的，所以能够寻找简单的规律进行计算。

网友 Winsty 给出了一个简捷的算法[1]，当 m 或 n 为偶数时，最短路长度为 mn；当 m 和 n 同时为奇数时，最短路长度为 mn 再加 0.41，因为要走一个斜边，如图 1-2 所示。



（a）m 或 n 为偶数　　（b）m 和 n 同时为奇数

图 1-2　Gridland 栅格与二维栅格大小的关系

【程序代码】

程序名称：	zju1037.c
题　　目：	Gridland
提交语言：	C
运行时间：	00:00.02
运行内存：	392K

```
#include <stdio.h>

int main()
{
    int iCase;                                  //测试例数
    scanf("%d", &iCase);
    int number;
    for (number=1; number<=iCase; number++)
    {
        int N, M;                               //二维栅格的大小
        scanf("%d%d", &M, &N);
        printf("Scenario #%d:\n", number);
        printf("%d.", M * N);
```

[1] http://acm.zjuwinsty.cn/post/105.html

```
            if(M%2 && N%2) printf("41");          //m和n同时为奇数
            else printf("00");                    //m或n为偶数
            printf("\n\n");
        }
        return 0;
    }
```

ZJU1045-HangOver[1、2、3]

Time limit: 1 Seconds Memory limit: 32768K

How far can you make a stack of cards overhang a table? If you have one card, you can create a maximum overhang of half a card length. (We're assuming that the cards must be perpendicular to the table.) With two cards you can make the top card overhang the bottom one by half a card length, and the bottom one overhang the table by a third of a card length, for a total maximum overhang of $1/2+1/3=5/6$ card lengths. In general you can make n cards overhang by $1/2+1/3+1/4+\cdots+1/(n+1)$ card lengths, where the top card overhangs the second by 1/2, the second overhangs the third by 1/3, the third overhangs the fourth by 1/4, etc., and the bottom card overhangs the table by $1/(n+1)$. This is illustrated in the figure below.

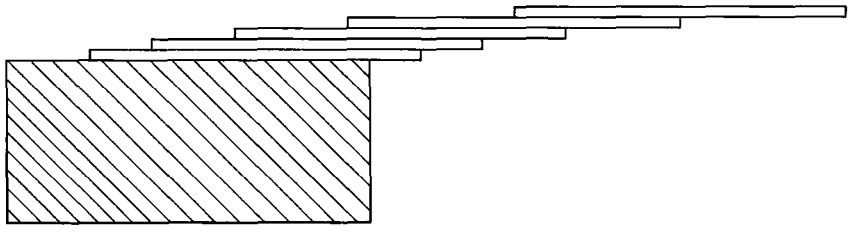


Figure 1-3

The input consists of one or more test cases, followed by a line containing the number 0.00 that signals the end of the input. Each test case is a single line containing a positive floating-point number c whose value is at least 0.01 and at most 5.20; c will contain exactly three digits.

For each test case, output the minimum number of cards necessary to achieve an overhang of at least c card lengths. Use the exact output format shown in the examples.

Example input

1.00

3.71

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1045

[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=1003

[3] http://acm.uva.es/archive/nuevoportal/data/problem.php?p=2294.

0.04
5.19
0.00

Example output

3 card(s)
61 card(s)
1 card(s)
273 card(s)

Problem Source

Mid-Central USA 2001

【题目大意】

你可以把一叠卡片放得离桌子多远？如果有一张卡片，那么可以达到的最远距离是卡片长度的一半。（假设卡片必须与桌子的边缘垂直。）使用两张卡片，使上面一张能放到的最远距离超过下面一张卡片长度的一半，而下面一张超出桌面的是卡片长度的 1/3，所以能达到的最远距离是 1/2＋1/3＝5/6 的卡片长度。

一般来说，n 张卡片能达到的最远距离是 1/2＋1/3＋1/4＋…＋1/(n＋1)，也就是最顶上的卡片超出第二张 1/2，第二张超出第三张 1/3，第三张超出第四张 1/4，等等，最后一张超出桌子的 1/(n＋1)。如图 1-3 所示。

输入有多组测试例，当一行是数字 0.00 时表示输入结束。每个测试例一行，是一个浮点数 c(0.01≤c≤5.20)，c 刚好是 3 位数字。

对每个测试例，输出达到距离 c 所需要的最少的卡片数量，输出格式如样例所示。

【算法分析】

题目给出卡片超过桌子边缘所能达到的最远距离 c，是一个浮点数，且保留 2 位小数。然后计算需要的最少的卡片数量 n。计算公式是：

$$\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n+1}\leqslant c$$

使用一个循环语句，逐渐累加，直到上式左边的和超过 c 时，就可以将 n 计算出来。

【程序代码】

程序名称：	zju1045.c
题　　目：	HangOver
提交语言：	C
运行时间：	00:00.00
运行内存：	388K

```
#include <stdio.h>

int main()
```

```
{
    //卡片超过桌子边缘所能达到的最远距离
    float c;
    while(scanf("%f", &c) && c != 0)
    {
        int i = 2;
        float fResult = 0;              //公式中左边的和
        while (fResult<c){
            fResult = fResult + 1.00 / i;
            i++;
        }
        //当左边的和 fResult 超过 c 时，输出结果（n=i-1，i 又多加了一次）
        printf("%d card(s)\n", i - 2);
    }
    return 0;
}
```

ZJU1049-I Think I Need a Houseboat[1、2、3]

Time limit: 1 Seconds Memory limit: 32768K

Fred Mapper is considering purchasing some land in Louisiana to build his house on. In the process of investigating the land, he learned that the state of Louisiana is actually shrinking by 50 square miles each year, due to erosion caused by the Mississippi River. Since Fred is hoping to live in this house the rest of his life, he needs to know if his land is going to be lost to erosion.

After doing more research, Fred has learned that the land that is being lost forms a semicircle. This semicircle is part of a circle centered at (0,0), with the line that bisects the circle being the x axis. Locations below the x axis are in the water. The semicircle has an area of 0 at the beginning of year 1. (Semicircle illustrated in the Figure.)

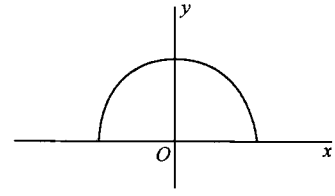


Figure 1-4

Input Format

The first line of input will be a positive integer indicating how many data sets will be included (N).

Each of the next N lines will contain the x and y Cartesian coordinates of the land Fred is considering. These will be floating point numbers measured in miles. The y coordinate will be non-negative. (0,0) will not be given.

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1049

[2] http://acm.pku.edu.cn/JudgeOnline/problem?id=1005

[3] http://acm.uva.es/archive/nuevoportal/data/problem.php?p=2363

Output Format

For each data set, a single line of output should appear. This line should take the form of:

"Property N: This property will begin eroding in year *z*."

Where *N* is the data set (counting from 1), and *z* is the first year (start from 1) this property will be within the semicircle AT THE END OF YEAR Z. Z must be an integer.

After the last data set, this should print out "END OF OUTPUT."

Notes:

1. No property will appear exactly on the semicircle boundary: it will either be inside or outside.

2. This problem will be judged automatically. Your answer must match exactly, including the capitalization, punctuation, and white-space. This includes the periods at the ends of the lines.

3. All locations are given in miles.

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 Source

Mid-Atlantic USA 2001

【题目大意】

弗雷德先生正考虑在路易斯安那州买一块地造房子。在土地调查中，他了解到由于密西西比河的侵蚀，路易斯安那州正在以每年 50 平方英里的速度变小。因为弗雷德先生希望在他的新房子里生活直至终老，所以他想知道他的那块地是否会被侵蚀掉。

经过进一步研究，弗雷德发现将要被侵蚀的土地呈半圆形。半圆是一个以（0，0）点为中心的圆的一半，半圆的直边是 x 轴。x 轴以下的部分在水中。第一年开始时，圆的面积是 0（半圆如图 1-4 所示）。

输入：

第一行是一个正整数 N，表示有几组测试数据。

接下来 N 行，每行是（x，y）坐标，弗雷德正考虑的地皮位置，坐标值是以英里为单位的浮点数，坐标 y 不会为负数，这两个数不会都为 0。

输出：

对每组测试数据，输出一行，其格式为：

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

其中 N 是测试数据编号（从 1 开始），经过 Z 年后，该处房产将位于半圆内。即经过 Z 年后，弗雷德的房子会被淹没。Z 必须是一个整数。

在所有测试数据之后，输出“END OF OUTPUT.”

【算法分析】

知道房产的位置坐标 (x, y)，就可以计算圆的半径 r：

$$r^2 = x^2 + y^2$$

半圆的面积：

$$\text{area} = \pi r^2/2$$

处于侵蚀范围的年数：

year=(int)ceil(area / 50.0)

其中函数 ceil(double x)，是得到大于 x 的最小整数。

【程序代码】

程序名称：	zju1049.c
题　　目：	I Think I Need a Houseboat
提交语言：	C
运行时间：	00:00.00
运行内存：	392K

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

const double PI = 3.1415927;                    //π的值不能取3.14

int main()
{
    int n, i = 1;
    int year;
    double x, y, radius, area;
    scanf("%d", &n);
    while(n--)
    {
        scanf("%lf%lf", &x, &y);
        radius = x * x + y * y;                 //计算r²
        area = PI * radius / 2.0;               //计算面积
        year = (int)ceil(area / 50.0);          //计算年数，注意函数ceil()
        printf("Property %d: ", i++);
        printf("This property will begin eroding in year %d.\n", year);
    }
    printf("END OF OUTPUT.\n");
    return 0;
}
```

ZJU1058-Currency Exchange[1、2]

Time limit: 1 Seconds Memory limit: 32768K

When Issac Bernand Miller takes a trip to another country, say to France, he exchanges his US dollars for French francs. The exchange rate is a real number such that when multiplied by the number of dollars gives the number of francs. For example, if the exchange rate for US dollars to French francs is 4.81724, then 10 dollars is exchanged for 48.1724 francs. Of course, you can only get hundredth of a franc, so the actual amount you get is rounded to the nearest hundredth. (We'll round .005 up to .01.) All exchanges of money between any two countries are rounded to the nearest hundredth.

Sometimes Issac's trips take him to many countries and he exchanges money from one foreign country for that of another. When he finally arrives back home, he exchanges his money back for US dollars. This has got Issac thinking about how much if his unspent US dollars is lost (or gained!) to these exchange rartes. You'll compute how much money Issac ends up with if he exchanges it many times. You'll always start with US dollars and you'll always end with US dollars.

Input

The first 5 lines of input will be the exchange rates between 5 countries, numbered 1 through 5. Line i will five the exchange rate from country i to each of the 5 countries. Thus the j entry of line i will give the exchange rate from the currency of country i to the currency of country j. the exchange rate form country i to itself will always be 1 and country 1 will be the US. Each of the next lines will indicate a trip and be of the form

$$N\ c1\ c2\ \cdots\ cn\ m$$

Where $1<=n<=10$ and $c1,\cdots, cn$ are integers from 2 through 5 indicating the order in which Issac visits the countries. (A value of $n=0$ indicates end of input, in which case there will be no more numbers on the line.) So, his trip will be $1\rightarrow c1\rightarrow c2\rightarrow\cdots\rightarrow cn\rightarrow 1$. the real number m will be the amount of US dollars at the start of the trip.

Output

Each trip will generate one line of output giving the amount of US dollars upon his return home from the trip. The amount should be fiven to the nearest cent, and should be displayed in the usual form with cents given to the right of the decimal point, as shown in the sample output. If the amount is less than one dollar, the output should have a zero in the dollars place.

This problem contains multiple test cases!

The first line of a multiple input is an integer N, then a blank line followed by N input blocks.

[1] http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1058

[2] http://acm.tju.edu.cn/toj/showp1193.html

Each input block is in the format indicated in the problem description. There is a blank line between input blocks.

The output format consists of *N* output blocks. There is a blank line between output blocks.

Sample Input

```
1
1 1.57556 1.10521 0.691426 7.25005
0.634602 1 0.701196 0.43856 4.59847
0.904750 1.42647 1 0.625627 6.55957
1.44616 2.28059 1.59840 1 10.4843
0.137931 0.217555 0.152449 0.0953772 1
3 2 4 5 20.00
1 3 100.00
6 2 3 4 2 4 3 120.03
0
```

Sample Output

```
19.98
99.99
120.01
```

Problem Source

East Central North America 2001, Practice

【题目大意】

当伊萨克·伯纳德·米勒（Issac Bernand Miller）到另一个国家去旅行时，比如去法国，他要用美元兑换法郎。汇率是一个实数，要兑换的美元乘以汇率即所得的法郎数。例如，若美元比法郎的汇率是 4.81724，则 10 美元可以换 48.1724 法郎。当然，你能得到的法郎数只能以百分之一为最小单位，所以实际得到的货币数要四舍五入到百分位（我们将 0.005 四舍五入为 0.01）。所有两国之间交换的货币，都要四舍五入到百分位。

有时，伊萨克要去许多国家旅行，他就要把一个国家的货币兑成另一个国家的货币。当他最终返家时，他就要把钱换回美元。这使伊萨克思索：对于他还没有花的钱，在兑换货币过程中，因为汇率不同，他会损失（或赢得）多少美元。你要计算出伊萨克在多次兑换货币后最终有多少钱。兑换过程从美元开始，最后又换回美元。

输入：

前 5 行数据是 5 个国家之间的汇率，从 1～5 编号。第 i 行的 5 个汇率，是第 i 个国家依次与 5 个国家之间货币的汇率。这样第 i 行的第 j 个数据就是从第 i 个国家到第 j 个国家的货币汇率。第 i 个国家到它自身的汇率总是 1，第 1 个国家是美国。

其后的每一行表示一次旅行，格式是：

$$N\ c1\ c2\ \cdots\ cn\ m$$

其中 $1 \leqslant n \leqslant 10$；$c1$，…，$cn$ 是从 2～5 的整数，表示伊萨克参观国家的顺序（$n=0$ 表示输入结束，这一行上就没有其他数字）。所以，他的旅程将是 $1 \rightarrow c1 \rightarrow c2 \rightarrow \cdots \rightarrow cn \rightarrow 1$。实数 m