在笔者的机器上，输出如下：

```
0.9999990
0.2499998
0.2
```

换句话说，在不断减 1e-6 的过程中出现了误差，使得循环终止时 f 并不等于 1，而是比 1 小一点。在除以 4 保留 1 位小数时成了 0.2。如果不出现误差，正确答案应该是 0.25 四舍五入保留一位小数，即 0.3。**一道好的竞赛题目应避免这种情况出现**[①]，但作为竞赛选手来说，有一种方法可以缓解这种情况：加上一个 EPS 以后再输出。这里的 EPS 通常取一个比最低精度还要小几个 3 个数量级的小实数。例如，要求保留 3 位小数时取 EPS 为 1e-6。这只是个权宜之计，甚至有可能起到“反作用”（如正确答案真的是 0.499999），但在实践中很好用（毕竟正确答案是 0.499999 的情况比 0.5 要少很多）。

4.5.2　例题一览和习题

本章共有 6 道例题，如表 4-2 所示。除了最后两道题目比较复杂之外，**读者应熟练掌握前 4 道题目的程序写法**。当然，为了巩固基础，让后面的学习更加轻松，笔者强烈建议大家独立实现所有 6 道题目。

表 4-2　例题一览

类　别	题　号	题目名称（英文）	备　注
例题 4-1	UVa1339	Ancient Cipher	排序
例题 4-2	UVa489	Hangman Judge	自顶向下逐步求精法
例题 4-3	UVa133	The Dole Queue	子过程（函数）设计
例题 4-4	UVa213	Message Decoding	二进制；输入技巧；调试技巧
例题 4-5	UVa512	Spreadsheet Tracking	模拟；一题多解
例题 4-6	UVa12412	A Typical Homework （a.k.a Shi Xiong Bang Bang Mang）	综合练习

下面是一些习题。这些题目的综合性较强，部分题目还涉及一些专门知识（如中国象棋、莫尔斯电码、RAID），理解起来也需要一定时间。另外一些题目需要一些思考，否则无从入手编写程序。由于这些题目的挑战性，**在继续阅读之前只需完成其中的 3 道题目**。如果想达到更好的效果，最好是完成 3 道或更多的题目。

习题 4-1　象棋（Xiangqi, ACM/ICPC Fuzhou 2011, UVa1589）

考虑一个象棋残局，其中红方有 n（$2 \leqslant n \leqslant 7$）个棋子，黑方只有一个将。红方除了有一个帅（G）之外还有 3 种可能的棋子：车（R），马（H），炮（C），并且需要考虑“蹩马腿”（如图 4-4 所示）与将和帅不能照面（将、帅如果同在一条直线上，中间又不隔着任何棋子的情况下，走子的一方获胜）的规则。

输入所有棋子的位置，保证局面合法并且红方已经将军。你的任务是判断红方是否已

[①] 方法有两种：一是删除答案恰好处于“舍入交界口”的数据，二是允许选手输出和标准答案有少许出入。

经把黑方将死。关于中国象棋的相关规则请参见原题。

习题 4-2　正方形（Squares, ACM/ICPC World Finals 1990, UVa201）

有 n 行 n 列（$2 \leqslant n \leqslant 9$）的小黑点，还有 m 条线段连接其中的一些黑点。统计这些线段连成了多少个正方形（每种边长分别统计）。

行从上到下编号为 1~n，列从左到右编号为 1~n。边用 H i j 和 V i j 表示，分别代表边(i,j)-(i,j+1)和(i,j)-(i+1,j)。如图 4-5 所示最左边的线段用 V 1 1 表示。图中包含两个边长为 1 的正方形和一个边长为 2 的正方形。

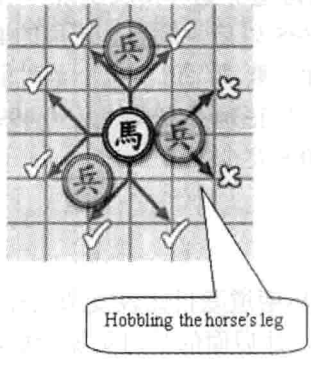

图 4-4　“蹩马腿”情况

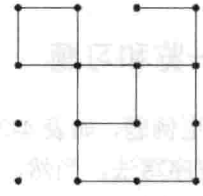

图 4-5　正方形

习题 4-3　黑白棋（Othello, ACM/ICPC World Finals 1992, UVa220）

你的任务是模拟黑白棋游戏的进程。黑白棋的规则为：黑白双方轮流放棋子，每次必须让新放的棋子“夹住”至少一枚对方棋子，然后把所有被新放棋子“夹住”的对方棋子替换成己方棋子。一段连续（横、竖或者斜向）的同色棋子被“夹住”的条件是两端都是对方棋子（不能是空位）。如图 4-6（a）所示，白棋有 6 个合法操作，分别为(2,3),(3,3),(3,5),(6,2),(7,3),(7,4)。选择在(7,3)放白棋后变成如图 4-6（b）所示效果（注意有竖向和斜向的共两枚黑棋变白）。注意(4,6)的黑色棋子虽然被夹住，但不是被新放的棋子夹住，因此不变白。

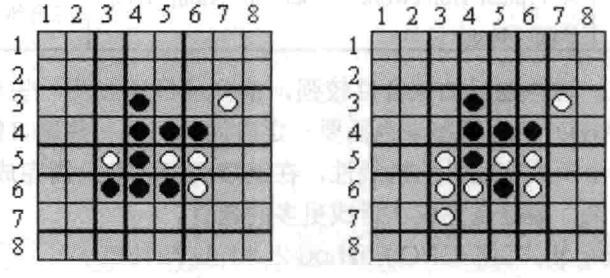

图 4-6　黑白棋

输入一个 8*8 的棋盘以及当前下一次操作的游戏者，处理 3 种指令：

- L 指令打印所有合法操作，按照从上到下，从左到右的顺序排列（没有合法操作时输出 No legal move）。
- Mrc 指令放一枚棋子在(r,c)。如果当前游戏者没有合法操作，则是先切换游戏者再

操作。输入保证这个操作是合法的。输出操作完毕后黑白方的棋子总数。

- Q 指令退出游戏，并打印当前棋盘（格式同输入）。

习题 4-4　骰子涂色（Cube painting, UVa 253）

输入两个骰子，判断二者是否等价。每个骰子用 6 个字母表示，如图 4-7 所示。

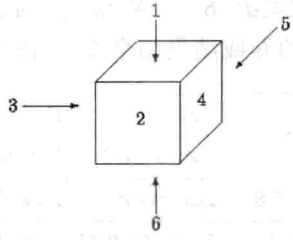

图 4-7　骰子涂色

例如 rbgggr 和 rggbgr 分别表示如图 4-8 所示的两个骰子。二者是等价的，因为图 4-8（a）所示的骰子沿着竖直轴旋转 90° 之后就可以得到图 4-8（b）所示的骰子。

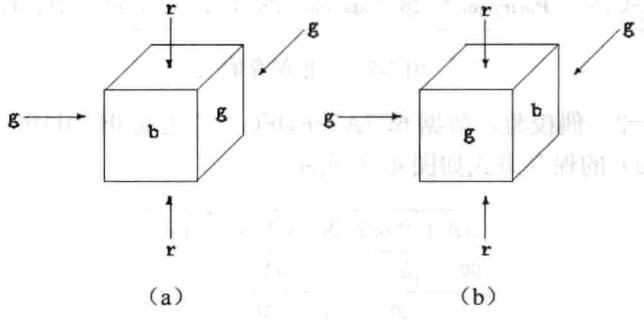

图 4-8　旋转前后的两个骰子

习题 4-5　IP 网络（IP Networks, ACM/ICPC NEERC 2005, UVa1590）

可以用一个网络地址和一个子网掩码描述一个子网（即连续的 IP 地址范围）。其中子网掩码包含 32 个二进制位，前 32−n 位为 1，后 n 位为 0，网络地址的前 32−n 位任意，后 n 位为 0。所有前 32−n 位和网络地址相同的 IP 都属于此网络。

例如，网络地址为 194.85.160.176（二进制为 11000010|01010101|10100000|10110000），子网掩码为 255.255.255.248（二进制为 11111111|11111111|11111111|11111000），则该子网的 IP 地址范围是 194.85.160.176~194.85.160.183。输入一些 IP 地址，求最小的网络（即包含 IP 地址最少的网络），包含所有这些输入地址。

例如，若输入 3 个 IP 地址：194.85.160.177、194.85.160.183 和 194.85.160.178，包含上述 3 个地址的最小网络的网络地址为 194.85.160.176，子网掩码为 255.255.255.248。

习题 4-6　莫尔斯电码（Morse Mismatches, ACM/ICPC World Finals 1997, UVa508）

输入每个字母的 Morse 编码，一个词典以及若干个编码。对于每个编码，判断它可能是哪个单词。如果有多个单词精确匹配，任选一个输出并且后面加上“!”；如果无法精确匹配，可以在编码尾部增加或删除一些字符以后匹配某个单词（增加或删除的字符应尽量少）。如果有多个单词可以这样匹配上，任选一个输出并且在后面加上“?”。

莫尔斯电码的细节参见原题。

习题4-7 RAID技术（RAID!, ACM/ICPC World Finals 1997, UVa509）

RAID技术用多个磁盘保存数据。每份数据在不止一个磁盘上保存，因此在某个磁盘损坏时能通过其他磁盘恢复数据。本题讨论其中一种RAID技术。数据被划分成大小为s（$1 \leqslant s \leqslant 64$）比特的数据块保存在$d$（$2 \leqslant d \leqslant 6$）个磁盘上，如图4-9所示，每$d$-1个数据块都有一个校验块，使得每$d$个数据块的异或结果为全0（偶校验）或者全1（奇校验）。

Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
Parity for 1-4	Data block 1	Data block 2	Data block 3	Data block 4
Data block 5	**Parity for 5-8**	Data block 6	Data block 7	Data block 8
Data block 9	Data block 10	**Parity for 9-12**	Data block 11	Data block 12
Data block 13	Data block 14	Data block 15	**Parity for 13-16**	Data block 16
Data block 17	Data block 18	Data block 19	Data block 20	**Parity for 17-20**
Parity for 21-24	Data block 21	Data block 22	Data block 23	Data block 24
Data block 25	**Parity for 25-28**	Data block 26	Data block 27	Data block 28

图4-9 数据保存情况

例如，d=5，s=2，偶校验，数据6C7A79EDFC（二进制01101100 01111010 01111001 11101101 11111100）的保存方式如图4-10所示。

Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
00	01	10	11	00
01	**10**	11	10	10
01	11	**01**	10	01
11	10	11	**11**	01
11	11	11	00	**11**

图4-10 数据6C7A79EDPC的保存方式

其中加粗块是校验块。输入d、s、b、校验的种类（E表示偶校验，O表示奇校验）以及b（$1 \leqslant b \leqslant 100$）个数据块（其中“?”表示损坏的数据），你的任务是恢复并输出完整的数据。如果校验错或者由于损坏数据过多无法恢复，应报告磁盘非法。

提示： 本题是位运算的不错练习，但如果没有RAID的知识背景，上述简要翻译可能较难理解，细节建议参考原题。

习题4-8 特别困的学生（Extraordinarily Tired Students, ACM/ICPC Xi'an 2006, UVa12108）

课堂上有n个学生（$n \leqslant 10$）。每个学生都有一个“睡眠-清醒”周期，其中第i个学生醒A_i分钟后睡B_i分钟，然后重复（$1 \leqslant A_i, B_i \leqslant 5$），初始时第$i$个学生处在他的周期的第$C_i$分钟。每个学生在临睡前会察看全班睡觉人数是否严格大于清醒人数，只有这个条件满足时才睡觉，否则就坚持听课A_i分钟后再次检查这个条件。问经过多长时间后全班都清醒。

如果用(A,B,C)描述一些学生，则图 4-11 中描述了 3 个学生(2,4,1)、(1,5,2)和(1,4,3)在每个时刻的行为。

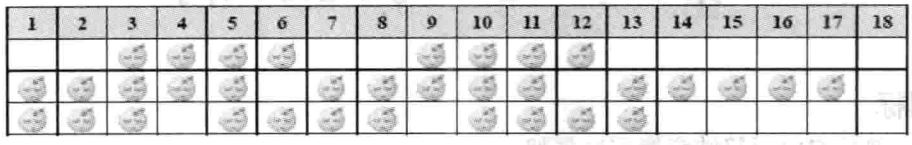

图 4-11　3 个学生每个时刻的行为

注意：有可能并不存在“全部都清醒”的时刻，此时应输出−1。

习题 4-9　数据挖掘（Data Mining, ACM/ICPC NEERC 2003, UVa1591）

有两个 n 元素数组 P 和 Q。P 数组每个元素占 S_P 个字节，Q 数组每个元素占 S_Q 个字节。有时需直接根据 P 数组中某个元素 $P(i)$的偏移量 $P_{ofs}(i)$算出对应的 $Q(i)$的偏移量 $Q_{ofs}(i)$。当两个数组的元素均为连续存储时 $Q_{ofs}(i)=P_{ofs}(i)/S_P*S_Q$，但因为除法慢，可以把式子改写成速度较快的 $Q_{ofs}(i)=(P_{ofs}(i)+P_{ofs}(i)<<A)>>B$。为了让这个式子成立，在 P 数组仍然连续存储的前提下，Q 数组可以不连续存储（但不同数组元素的存储空间不能重叠）。这样做虽然会浪费一些空间，但是提升了速度，是一种用空间换时间的方法。

输入 n、S_P 和 S_Q（$N\leqslant 2^{20}$，$1\leqslant S_P$，$S_Q\leqslant 2^{10}$），你的任务是找到最优的 A 和 B，使得占的空间 K 尽量小。输出 K、A、B 的值。多解时让 A 尽量小，如果仍多解则让 B 尽量小。

提示：本题有一定实际意义，不过描述比较抽象。如果对本题兴趣不大，可以先跳过。

习题 4-10　洪水！（Flooded! ACM/ICPC World Finals 1999, UVa815）

有一个 $n*m$（$1\leqslant m$，$n<30$）的网格，每个格子是边长 10 米的正方形，网格四周是无限大的墙壁。输入每个格子的海拔高度，以及网格内雨水的总体积，输出水位的海拔高度以及有多少百分比的区域有水（即高度严格小于水平面）。

本题有多种方法，能锻炼思维，建议读者一试。

4.5.3　小结

指针还有很多相关内容本书没有介绍，例如，指向 void 型的指针、指向函数的指针、指向常量的指针以及指针和数组之间的关系（注意，尽管在很多地方可以混用，但指针和数组不是一回事！《C 语言程序设计奥秘》用一章的篇幅来叙述二者的区别）。正如书中所说，本书将尽量回避指针，但尽管如此，调试并理解前面几个 swap 函数的工作方式对于理解计算机的工作原理大有好处。

递归需要从概念和语言两个方面理解。从概念上，递归就是“自己使用自己”的意思。递归调用就是自己调用自己，递归定义就是自己定义自己……当然，这里的“使用自己”可以是直接的，也可以是间接的。很多初学者在学习递归时专注于表象，从而未能透彻理解其“计算机”本质。由于我们的重点是设计算法和编写程序，理解递归函数的执行过程是非常重要的。因此，本章大量使用了 gdb 作为工具讲解内部机理，即使读者在平时编程时不用 gdb 调试，在学习初期用它帮助理解也是大有裨益的。关于 gdb 的更多介绍参见附录 A。