2001 年中山大学第四届大学生程序设计竞赛

暨第 26 届 ACM/ICPC 中山大学组队选拔赛

中山大学教务处主办

中山大学信息科学与技术学院计算机科学系实施
中山大学信息学研究实验室命题

中山大学，广州/中山大学珠海校区，珠海

第一轮竞赛

2001 年 5 月 12 日

这份题目应当有 8 页，共 5 题。

如果发现缺页请立即联系竞赛工作人员。

注意事项

欢迎参加2001年中山大学第四届大学生程序设计竞赛暨第26届ACM/ICPC中山大学组队选拔赛。为了能够更好地享受竞赛的乐趣，请注意以下事项：

1. 竞赛时间4小时，共5道题。

2. 遵守竞赛的规则。请不要携带任何书籍资料、计算器和计算器可读的媒体（如磁盘）进场。竞赛结束后，请遵从竞赛工作人员的指示离场。

3. 你可以按照自己喜欢的顺序解题，不必遵守题目实际出现的顺序。你也可以按照自己喜欢的风格编写程序。在竞赛中不需要写任何文档。

4. 请将源程序编译成可执行文件，注意可执行文件名应符合题目的要求。不要使用非标准的库（另外，使用Pascal的选手请注意不要使用CRT单元，使用C/C++的选手请注意不要使用conio.h）。如果你在编译时有任何困难，请立即联系竞赛工作人员。竞赛结束后，请不要对源程序作任何改动。

5. 请按照题目的要求输入输出。

   • 不要自行更改输入输出的方式。对采用文件输入输出的题目，请从指定的文本文件中读取数据，输出到指定的文本文件，程序在执行过程中不需要与人进行交互。文件名中不要包含驱动器符号和绝对路径，你可以认为输入输出文件与可执行文件在同一个目录下。在竞赛期间，请使用文本编辑器自行创建输入文件。

   • 在数据输入时，请不要更改数据输入的格式、次序、换行要求和数据的分隔字符，也不要输出任何的提示信息，即使这样的更改可能有利简化程序。除了题目声明之外，请不要对数据的范围作任何的假设，你可以认为所有的输入都符合题目规范，不需要编程处理不符合题目规范的输入。

   • 请严格按照题目要求的格式输出。程序的输出的格式应当与样例输出的格式完全一致。包括对单词拼写（包括错误拼写）、字母大小写、标点、空格和换行的要求。请不要使用制表字符（Tab）。除了题目要求之外，不要输出多余空行，或在行末输出多余的空格，也不要连续输出多个空格。

6. 在竞赛期间，每隔一定时间请自行将源程序存盘备份，避免机器故障带来麻烦。

7. 竞赛评委的决定是最终的。

8. 如果你发现机器有问题或对竞赛有任何疑问，请立即联系竞赛工作人员。

第一题 识别浮点常量（40 分）

可执行文件：identify.exe
输入方式：键盘输入
输出方式：屏幕输出

编译器在对程序进行编译之前，首先要进行语法分析。通常，程序被分解成若干个小单元，然后和语言的语法模式进行匹配。在分析表达式的时候，变量的类型在变量声明的时候就决定了；而常量的类型需要从常量的形式来判断。

假设你是自动编译机器（ACM）开发小组的一员，负责 Pascal 语言编译器的开发。你的任务是分析程序分解模块送来的文件，判断其中包含的字符串是否合乎语法的 Pascal 浮点常量。

Pascal 语言对浮点常量的语法要求是：一个浮点常量除了十进制数码之外，必须带有一个小数点或/和一个指数（紧接在字母 e 或 E 之后，在正式文档中也被称为比例因子）。如果该浮点常量含有小数点，则在小数点两侧都至少要有一个十进制数码。当然，在整个浮点常量或/和指数之前，也许会出现符号+或-。指数不能包含小数。空格也许会出现在浮点常量的前后，但不会出现在浮点常量中间。

请注意 Pascal 语言的语法规则没有对浮点常量的取值范围做出任何假定。

输入

输入只有一行，就是有待识别的字符串。字符串的长度不超过 255。

输出

请将分析的结果按以下样例的格式输出。如果输入文件中的字符串是 Pascal 浮点常量，请输出字符串“YES”，否则输出字符串“NO”。

输入样例	输出样例
1.2	YES
1.	NO
1.0e-55	YES
e-12	NO
1e-12	YES
6.5E	NO
+4.1234567890E-99999	YES

第二题 版本大混乱？（40 分）

可执行文件：versions.exe
输入方式：键盘输入
输出方式：屏幕输出

计算机软件是生命期最短的消费品之一。对商业软件来说，每年发布一个新的软件版本看起来是合适的。在版本 1.0 之后，我们有版本 2.0，接着是版本 2.1，3.0，等等。公共软件，例如 GNU 软件，或 Liunx，它们更新生命周期甚至更短——可能每天都有一个新版本出现。

版本号通常被用来区分一种软件包在不同时间的发布。大部分软件版本号都是用“.”分隔的非负整数的序列。对两个不同的版本 $A=a_1.a_2.\cdots.a_n$ 和 $B=b_1.b_2.\cdots.b_m$，如果下面两个条件之一成立，我们认为版本 A 要比 B 新：

1．对某个 i，我们有：对所有 $j<i$，$a_i>b_i$ 和 $a_j=b_j$；

2．n 比 m 大，而且对所有 $i<m$，$a_i=b_i$。

在这个问题里，你要对给定的一组版本号按照上面的定义从旧到新排序。

输入

输入包含若干行，第一行是一个整数 n（n<=20），表示要排序的版本数。接下来的 n 行，每行包含一个版本号。每个版本号是一个长度不超过 50 的字符串。你可以认为版本号都符合上面的定义。

输出

将排序的结果按以下样例的格式输出。

输入样例	输出样例
4 3.0.5 1 2.4 2.4.6	1 2.4 2.4.6 3.0.5

第三题 锦标赛（40 分）

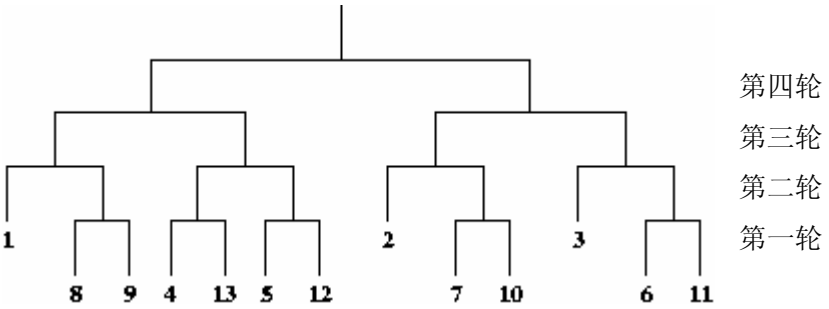
可执行文件：seeding.exe
输入方式：键盘输入
输出方式：屏幕输出

在许多竞赛当中，选手们被赋予种子排名代表他们的相对实力，最优秀那个选手的种子排名是 1，其次那个选手的种子排名是 2，依次类推。在安排一个淘汰赛制的锦标赛赛程时，总是要求最优秀的选手尽可能迟点碰头（例如，直到最后一轮比赛才有可能看到 1 号种子和 2 号种子比赛）。

为了说明赛程的安排规则，我们来定义两个名词。一场比赛的“实力”是比赛双方选手的种子排名之和。假如最好的两个选手都能顺利晋级而在某场比赛碰头，我们就称该场比赛的“实力”是“最佳实力”（换句话说，一场比赛的“最佳实力”就是可能参加该场比赛的

两个选手的种子排名之和的最小值)。

一个有 n 名选手参加的锦标赛总共需要进行 $r=\lceil \log_2 n \rceil$ 轮比赛。如果我们称决赛是第 r 轮，半决赛是第 r-1 轮，…，则赛程的安排规则可以描述如下：在第 k 轮的所有比赛的“最佳实力”都是 $2^{r-k+1}+1$。举例来说，一个有 13 名选手参加的锦标赛赛程如下图所示：



你的任务是解决下面的问题：给出 n 和 m 的值，计算在一个有 n 名选手参加的锦标赛当中，实力为 m 的比赛最早可能出现在第几轮。你可以假定赛程就是按照上面所描述的方式安排的。

输入

输入只有一行，包含两个用空格分隔的整数 n 和 m，其中 $2 \leqslant n \leqslant 100$，$m \geqslant 3$。你可以假设实力为 m 的比赛一定可能在某轮出现。

输出

请将计算的结果按以下样例的格式输出。

输入样例	输出样例
100 3	7
13 10	2

第四题 随机数（40 分）

可执行文件：random.exe
输入文件：random.in
输出文件：random.out

在这个问题里，“黑箱”是一个简单的数据库。它可以保存一组整数，同时有一个特殊的变量 i，初始时黑箱是空的，而且 i 的值是 0。这个黑箱能够接收操作指令序列并进行处理。有两种操作指令：

- ADD(x)：将整数 x 放入黑箱；

- GET：将 i 加 1 并输出黑箱所有整数中第 i 小的元素。所谓第 i 小的元素，是将黑箱中的元素按非递减排序后，在第 i 个位置上的数。

我们来看看一个有 11 条操作指令的例子：

N	操作指令	i	执行指令后黑箱中的内容（元素按非递减排序）	输出
1	ADD(3)	0	3	
2	GET	1	**3**	3
3	ADD(1)	1	1,3	
4	GET	2	1,**3**	3
5	ADD(-4)	2	-4,1,3	
6	ADD(2)	2	-4,1,2,3	
7	ADD(8)	2	-4,1,2,3,8	
8	ADD(-1000)	2	-1000,-4,1,2,3,8	
9	GET	3	-1000,-4,**1**,2,3,8	1
10	GET	4	-1000,-4,1,**2**,3,8	2
11	ADD(2)	4	-1000,-4,1,2,2,3,8	

我们用两组整数去描述操作指令序列：

1. A(1),A(2),…,A(M)：输入黑箱的元素序列，A 的值是绝对值不大于 2,000,000,000 的整数，M≤30000。对应上面的例子，我们有 A=(3,1,-4,2,8,-1000,2)。

2. u(1),u(2),…,u(N)：一个自然数序列，表示第一个，第二个，…，第 N 个 GET 操作指令执行时黑箱中的元素个数。对应上面的例子，我们有 u=(1,2,6,6)。

黑箱的算法要求自然数序列 u(1),u(2),…,u(N)是非递减的，N≤M，并且对每个 p（1≤p≤N）不等式 p≤u(p)≤M 总是成立的。因为序列 u 中的第 p 个元素代表要执行一个 GET 操作输出 A(1),A(2),…,A(u(p))当中第 p 小的元素。

在为黑箱算法设计测试数据的时候我们遇到了下面的问题。普通的随机数发生器并不适合用来产生随机序列{u(i)}。因为序列本身附加了某种限制，如果采用逐个生成序列中的元素，每次选择的随机数都符合限制的方法，产生的序列从整体上说并不是完全随机的。

我们发现这个问题可以用下面的方法解决。如果我们按字典序枚举所有满足要求的序列并为每一个序列编号，只要我们产生一个随机数，对应的序列就相当于一个随机的序列。乍一看好象只要写一个程序按字典序产生所有满足要求的序列就足够了。仔细想想！即使 M 和 N 的值不是很大，要穷举这些序列也可能要花最先进的计算器几个世纪的时间。但是只要我们能够根据编号立即产生所需的序列，就可以避免穷举所有序列。

不过这仅仅是问题的一部分。因为序列的数量实在太多了，序列的编号也许是一个很大的数字，甚至可能有上百位，远远超出普通的随机数发生器所能产生随机数的范围。于是我们决定由一个[0,1]区间的随机实数产生一个很大的随机整数。具体来说，将这个随机实数用二进制表示成：$0.b_1b_2\cdots$，所有的 b_i=0 或 1。然后用下面的公式将这样一个实数与[A,B]区间的一个整数联系起来：

$G(A,B,0.b_1b_2\cdots b_p)$ =	p=0 或 A=B：A 否则： b_1=0：$G(A,(A+B)\ \mathrm{div}\ 2,0.b_2b_3\cdots b_p)$ b_1=1：$G((A+B)\ \mathrm{div}\ 2,B,0.b_2b_3\cdots b_p)$

其中 A≤B，p≥0，“div”表示整除。

给定 M，N（1≤N≤M≤200）和一个二进制实数 $0.b_1b_2\cdots b_p$（1≤p≤400），你的任务是写一个程序找出 $G(1,T,0.b_1b_2\cdots b_p)$ 对应的 u 序列。更精确的说，令 T 表示对给定的 M 和 N 所有满足要求的 u 序列总数，将这 T 个序列按字典序从 1 开始编号，你的程序应找出编号为 $G(1,T,0.b_1b_2\cdots b_p)$ 的 u 序列。下面的例子按字典序列出了 M=4 和 N=3 时所有符合要求的序列。

```
1, 2, 3          （这里 T=14）
1, 2, 4
1, 3, 3
1, 3, 4
1, 4, 4
2, 2, 3
2, 2, 4
2, 3, 3
2, 3, 4
2, 4, 4
3, 3, 3
3, 3, 4
3, 4, 4
4, 4, 4
```

输入

输入文件 random.in 有两行，第一行是两个用空格分隔的整数 M 和 N，第二行包含二进制实数 $0.b_1b_2\cdots b_p$，这一行没有空格。

输出

请将对应的序列 u(1),u(2),…,u(N)输出到文件 random.out。输出的数字用空格和/或回车分隔。

输入样例	输出样例
4 3 0.01101101011110010001101010001011010	2 2 4

第五题 推箱子游戏（40 分）

可执行文件：pushing.exe

输入文件：pushing.in

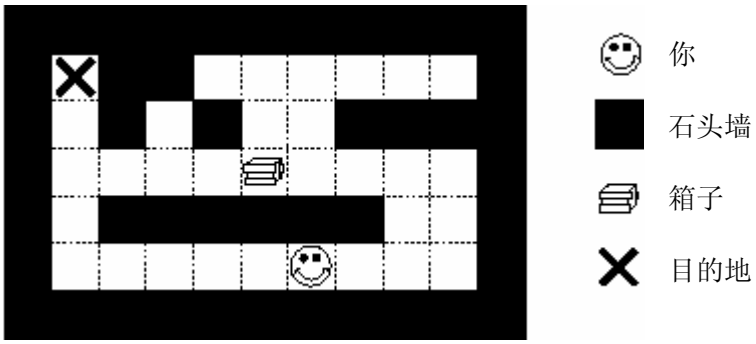
输出文件：pushing.out

你玩过“推箱子”，或者称为“仓库管理员”的游戏吗？如果没玩过，不要紧，下面我来解释一下这种游戏的玩法。

试想象你站在一个二维的仓库里，这个仓库由方格组成，每个方格或者是石头墙，或者是空的。你可以向东、南、西或北一次走一格。像这样的移动称为“走”。

在某个空格中有一个箱子，它可以移到相邻的空格中，方法是你站在箱子的另一侧的空格去推它。当你推动箱子以后，你就移动到箱子本来的位置。像这样的移动称为“推”。这个箱子无法用推以外的方法移动。这就意味着如果你把箱子推到一个角落里，你就再也没有办法把它移出这个角落了。

一个空格被标记为目的地。你的工作是将箱子推到目的地，推的过程可以用“走”和“推”的序列来描述。因为箱子很重，你想尽量使推的次数最少。你能写一个程序找出最佳的方案吗？



输入

输入文件 pushing.in 有若干行，第一行是两个用空格分隔的整数 r 和 c（1≤r,c≤20），分别表示仓库的行数和列数。接下来的 r 行，每行包含 c 个字符，每个字符代表仓库的一格。一格石头墙用“#”表示，一个空格用“.”表示。你的初始位置用“S”表示，箱子的初始位置用“B”表示，目的地用“T”表示。

输出

请将“走”和“推”的序列输出到文件 pushing.out。输出的序列应当使推的的次数最少。如果有多个符合题目要求的序列，输出一个使总移动步数最少（“走”和“推”）的序列。如果仍有多个序列，你可以选择任意一个输出。

“走”和“推”的序列是用由字母 E、S、W、N、e、s、w 和 n 组成的字符串。大写字母表示“推”，小写字母表示“走”。不同的字母分别表示东、南、西和北。

如果不可能将箱子推到目的地，请输出“Impossible.”

输入样例	输出样例
1 7 SB....T	EEEEE

1 7 SB..#.T	Impossible.
7 11 ########### #T##......# #.#.#..#### #....B....# #.######..# #.....S...# ###########	eennwwWWWWeeeeeesswwwwwwwnNN
8 4 .... .##. .#.. .#.. .#.B .##S .... ###T	swwwnnnnnneeesssSSS