猜数

guess

【问题背景】

佳佳和栋栋一起参加了在四川绵阳举行的全国信息学奥林匹克冬令营。今天是休息的日子，佳佳和栋栋闲着无聊，决定一起玩经典的猜数字游戏。

游戏的规则是这样的：

佳佳在心中默想一个十进制四位数 $a_1a_2a_3a_4$，且约定 $a_1\neq a_2\neq a_3\neq a_4$，$a_1$ 可为 0。栋栋首先猜一个十进制四位数 $b_1b_2b_3b_4$，当然 $b_1\neq b_2\neq b_3\neq b_4$，$b_1$ 可为 0。猜完后，佳佳马上将 $b_1b_2b_3b_4$ 与 $a_1a_2a_3a_4$ 作比较，并以整数 P 和整数 Q 来回答，其中 P 表示所猜的数字和位置都是对的数的个数，Q 表示不管位置对还是不对，所猜的数字与被猜的数字有几个数字是相同的。如果栋栋没有猜对，那么他还可以继续猜，直到猜对为止。栋栋的目标是使猜的次数尽量少。

例如，佳佳默想的数是 6703。

第一次，栋栋猜 1980，佳佳回答 P=0，Q=1

第二次，栋栋猜 0731，佳佳回答 P=1，Q=3

第三次，栋栋猜 6703，佳佳回答 P=4，Q=4

于是，游戏结束，栋栋只猜了三次就猜出了佳佳所默想的数。

当然，首发成功栋栋觉得很爽，不过后来栋栋就很难猜出佳佳所想的数了。栋栋想，能不能使用计算机来帮助自己用尽量少的次数猜出佳佳所想的数呢？经过几个小时在网上的紧张搜索，栋栋终于找到了佳佳在 y 年前所编写的猜数程序。正当栋栋乐滋滋的拿着程序准备再和佳佳玩猜数游戏时，佳佳笑嘻嘻的对栋栋说要改规则，提高难度。于是，规则被改成了猜长度为 $L(2\leqslant L\leqslant 9)$ 的数。这次，佳佳的程序没有起到任何作用，结果是栋栋花了十次猜了佳佳想的一个长度 L 为 1 的数，其他的都没有猜出来。

栋栋觉得很不爽，希望能找到新的程序，他把所有的希望都寄托在了你的身上。

栋栋已经请高手编写了“外壳”，你的任务是编写猜的过程，为了和“外壳”能兼容，你需要调用库函数来与用户交互。至于“外壳”是怎么工作的你不必关心。

这是一道交互式的试题。你的程序不能直接对文件进行操作，而需要调用库函数来完成相应的操作。系统提供一个数据类型 TNumber 和三个库函数 Len、tryNumber 和 Next，它们的说明和使用方法如下：

TNumber：一个最多含有 10 个字节的数组类型，用于函数 tryNumber 的参数传递；

Len：获取待猜数的长度 L 的函数，可以任意次调用；

tryNumber(N,P,Q)：其中 N 为 TNumber 类型，表示所猜的数，P,Q 为两个整形数。当调用这个函数后，库会根据 N 与待猜数的不同，给出 P 与 Q 的值并返回。

Next：为了评价你的程序，可能要它连续猜多个数，当猜完一个数后(只有在调用函数 tryNumber 后返回的 P,Q 值均等于 Len 时，才算猜完)，需要调用 Next

接着猜下一个数。如果让你猜的所有的数都猜完了，调用 Next 时，库会自动退出。

上述的数据类型和函数在 Pascal 和 C++中的定义如下：

类型/函数	Pascal	C++
TNumber	type TNumber = array[0..9] of byte;	typedef char TNumber[10];
Len	function Len: integer;	int Len();
tryNumber	procedure tryNumber(const N: TNumber; var P, Q: integer);	void tryNumber(const TNumber N, int &P, int &Q);
Next	procedure Next;	void Next();

如果你使用 Pascal 语言，你用语句

Uses GuessLib_p;

来引用我们提供的单元。

如果你使用 C++语言，请在工程中包含文件 GuessLib_c.o 并在程序头加一句

#include “GuessLib_c”

【如何测试你的程序】

你可以在你的目录下建立一个名为 guess.in 的文件，文件的第一行包含两个整数 n 和 L，分别表示要猜的数的个数和数的长度。接下来 n 行，每行 L 个 0 至 9 的互异的整数，用空格分隔，第 i+1 行表示的是第 i 个要猜的数的每一位。

为了方便你调试，我们在库中包含了随机产生待猜数的过程，如果长度 L 是负数，库将随机产生 n 个长为-L 的数，而不需要将这些数放入 guess.in 中。因此，输入文件中可以只包含两个整数 n 和 L'，分别表示待猜数的个数和长度的相反数。

当建立好输入文件后，就可以运行你所编写的程序。所有的中间结果将被放在 guess.log 中。一般包含这些信息：

- Game started.：游戏开始。
- The number is [<Number>]：当前要猜的数为<Number>
- <ID> Guess [<Number>] P=<P> Q=<Q>：调用了一次 tryNumber 函数，传入的参数为<Number>，得到 P 的返回值为<P>，Q 的返回值为<Q>，<ID>为调用的编号。
- <T> Times.：调用了 Next，猜对上一个数一共调用了<T>次 tryNumber。
- Finished. Average query: <Ave>：猜完了所有的数，平均使用了 Ave 次询问。
- Error: Number is invalid.：tryNumber 的参数是不合法的。
- Error: Invalid call of Next.：在没有猜出当前数的情况下就调用了 Next 函数。
- Error: Too many numbers.：输入文件中要猜的数太多了。
- Error: Input file error.：输入文件格式错误。

【一个成功交互的例子】

Guess.in

1 4
6 7 0 3

函数调用过程	运行情况	Guess.log
Len	返回值：4	Game started. The number is [6703] 1 Guess [1980] P = 0 Q = 1 2 Guess [0731] P = 1 Q = 3 3 Guess [6703] P = 4 Q = 4 3 Times. Finished. Average query: 3.00
tryNumber({1,9,8,0},P,Q)	说明：为书写简便，用{1,9,8,0}表示所传数组的前四个位置分别为1,9,8,0 其他位置上的值不重要 返回：P=0 Q=1	
tryNumber({0,7,3,1},P,Q)	返回：P=1 Q=3	
tryNumber({6,7,0,3},P,Q)	返回：P=4 Q=4	
Next	说明：所有数已猜完，结束 平均次数：3 次	

【评分方法】

对于每一个测试点， 我们会有很多个数需要你的程序猜，你必须在规定的时间内猜出所有的数。最后根据猜每个数所用的平均次数 Ave 与我们考前提供的两个得分参考值 *Good*, *Bad*(*Good*<*Bad*)按下面的公式计算你的得分：

$$Score=\begin{cases}10(Ave\le Good)\\ 2+\left\lfloor 8*\left(\dfrac{Bad-Ave}{Bad-Good}\right)^2\right\rfloor(Good<Ave<Bad)\\ 2(Ave\ge Bad)\end{cases}$$

其中 $\lfloor a\rfloor$ 表示取 a 的整数部分。

另外，如果你的程序出现以下情况之一，对应的测试点得 0 分：

- 无源程序或源程序编译不通过
- 读写任何文件
- 读写库函数或库变量的内存
- 函数调用不合法
- 使库异常退出
- 自行结束运行
- 空间、时间超过限制
- 其他使库没能正常产生平均次数并退出的情况

【测试点信息】

测试时，我们将有 10 个测试点：

测试点编号	1	2	3	4	5	6	7	8	9	10

Len	2	3	4	5	6	7	8	9	10	10
要猜的数的个数	100	100	100	100	100	20	20	20	1	20

注意：

1.同一个数据里所有数的长度都是一样的，所有要猜的数都是随机给出的。

2.第 9 个点的 *Bad* 和 *Good* 值会较大。