排列游戏（perm）

这是一道交互题。

【题目描述】

小 H 和小 L 正在玩一个猜排列的游戏。

小 H 有一个 $0 \sim n-1$ 的排列 $p = [p_0, p_1, \ldots, p_{n-1}]$。现在小 L 知道了排列的长度 n，他希望通过特定的询问方式猜出这个排列 p。具体地，小 L 可以向小 H 提出如下形式的询问：

- 给定非负整数 l, r 满足 $0 \le l \le r \le n-1$，求 $p_l, \ldots, p_r$ 中**未出现过的最小非负整数**。

不过小 H 和小 L 发现，即便可以进行任意多次询问，有时也不足以唯一确定这个排列 p。于是两人约定：假设小 H 的答案是 p，而小 L 猜测的排列是 q，如果在 p 和 q 这两个排列中，对于任意 $0 \le l \le r \le n-1$，区间 $p_l, \ldots, p_r$ 中未出现过的最小非负整数，始终等于区间 $q_l, \ldots, q_r$ 中未出现过的最小非负整数，那么就认为小 L 的猜测是正确的。

为了增加游戏的难度，小 H 限制了小 L 的询问次数。你需要帮助小 L 猜出小 H 的排列。

【实现细节】

选手不需要，也不应该实现 `main` 函数。

选手需要确保提交的程序包含头文件 `perm.h`，即在程序开头加入以下代码：

```
#include "perm.h"
```

选手需要在提交的程序源文件 `perm.cpp` 中实现以下两个函数：

```
void init(int c, int t);
```

- c, t 分别表示测试点编号与测试数据组数。$c = 0$ 表示该测试点为样例。
- 对于每个测试点，该函数会在**程序开始运行时**被交互库调用**恰好一次**。

```
std::vector<int> perm(int n);
```

- n 表示排列的长度。
- 该函数需要返回一个 $0 \sim n-1$ 的排列，表示小 L 的猜测。
- 对于每个测试点，该函数会被交互库调用**恰好** t **次**。

选手可以通过调用以下函数进行一次询问：

```
int query(int l, int r);
```

- l, r 表示询问的区间。选手需要保证 $0 \le l \le r \le n-1$。

- 该函数会返回 $p_l, \ldots, p_r$ 中未出现过的最小非负整数。
- 选手需要保证交互库每次调用 perm 时，调用该函数的次数不超过 6×10^5。

注意：在任何情况下，最终测试时所用的交互库运行所需时间均不会超过 0.1 秒，所用内存为固定大小，且均不超过 64 MiB。

【测试程序方式】

试题目录下的 grader.cpp 是交互库参考实现，最终测试时所用的交互库实现与该参考实现有所不同，因此选手的解法不应该依赖交互库实现。

选手可以在本题目录下使用如下命令编译得到可执行程序：

```
g++ grader.cpp perm.cpp -o perm -std=gnu++14 -O2 -static
```

对于编译得到的可执行程序：

- 可执行文件将从**标准输入**读入以下格式的数据：
  - 输入的第一行包含两个非负整数 c, t，分别表示测试点编号和测试数据组数。
  - 接下来依次为每组测试数据，对于每组测试数据：
    - 第一行包含一个正整数 n，表示排列的长度。
    - 第二行包含 n 个非负整数 $p_0, p_1, \ldots, p_{n-1}$，表示小 H 的排列。
- 可执行文件将输出以下格式的数据至**标准输出**：
  - 对于每组测试数据：
    - 第一行包含一个字符串，表示测试的结果。其中，
      - `Correct.` 表示选手返回的结果正确；
      - `Wrong answer.` 表示选手返回的结果错误；
      - `Invalid operation.` 表示选手对 query 函数的调用不合法。
    - 若结果为 `Correct.`，则第二行包含一个非负整数，表示选手在所有测试数据中调用 query 函数的次数的最大值。

【样例 1 输入】

```
0 1
6
4 2 3 5 0 1
```

【样例 1 输出】

```
Correct.
4
```

【样例 1 解释】

该样例共包含一组测试数据。

对于第一组测试数据，小 H 的排列为 $p = [4, 2, 3, 5, 0, 1]$。

以下是一种可能的交互过程：

- 调用 `query(0, 3)`，交互库返回 $4, 2, 3, 5$ 中未出现过的最小非负整数 0。
- 调用 `query(3, 4)`，交互库返回 $5, 0$ 中未出现过的最小非负整数 1。
- 调用 `query(1, 5)`，交互库返回 $2, 3, 5, 0, 1$ 中未出现过的最小非负整数 4。
- 调用 `query(3, 5)`，交互库返回 $5, 0, 1$ 中未出现过的最小非负整数 2。
- 返回 $q = [4, 2, 5, 3, 0, 1]$。

可以证明，排列 q 被认为是正确的。

【样例 2】

见选手目录下的 ***perm/perm2.in*** 与 ***perm/perm2.ans***。

该样例满足测试点 $4 \sim 8$ 的约束条件。

【下发文件说明】

在本试题目录下：

1. `grader.cpp` 是提供的交互库参考实现。
2. `perm.h` 是头文件，选手不用关心具体内容。
3. `template_perm.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

选手注意对所有下发文件做好备份。最终评测时只测试本试题目录下的 `perm.cpp`，对该程序以外文件的修改不会影响评测结果。

【数据范围】

对于所有测试数据，均有：

- $t = 10$；
- $2 \le n \le 3 \times 10^4$；
- 对于所有 $0 \le i \le n - 1$，均有 $0 \le p_i \le n - 1$，且 p 是 $0 \sim n - 1$ 的排列。

测试点编号	$n =$	特殊性质
$1 \sim 3$	10	无
$4 \sim 8$	10^2	
$9, 10$	3×10^4	A
$11, 12$		B
$13, 14$		C
$15 \sim 20$		无

特殊性质 A：$p_0 = 0$。

特殊性质 B：存在非负整数 $k \in [0, n-1]$ 满足 $p_0, \ldots, p_k$ 单调递减，且 $p_k, \ldots, p_{n-1}$ 单调递增。

特殊性质 C：p 在所有 $0 \sim n-1$ 的排列中**独立均匀随机**生成。

【评分方式】

注意：

- 选手不应当通过非法方式获取交互库的内部信息，如直接与标准输入、输出流进行交互。此类行为将被视为作弊；
- **最终的评测交互库与样例交互库的实现不同。**

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制等会导致相应测试点得 0 分等。选手只能在程序中访问自己定义的变量以及交互库给出的变量，尝试访问其他地址空间将可能导致编译错误或运行错误。

每次调用 `perm` 函数时，若返回的排列不被认为是正确的，或 `query` 函数调用不合法，或 `query` 函数调用次数超过 6×10^5，则相应测试点得 0 分。

在上述条件基础上：

- 在测试点 $1 \sim 3$ 中，程序得到满分当且仅当每次调用 `perm` 函数时，`query` 函数调用次数不超过 10^2。
- 在测试点 $4 \sim 8$ 中，设 m 表示每次调用 `perm` 函数时的询问次数的最大值，则程序将获得 $5 \cdot f(m)$ 分，其中 f 的计算方式如下表所示：

m	$f(m) =$
$m \le 100$	1
$100 < m \le 200$	$1 - \frac{\sqrt{m-100}}{50}$
$200 < m \le 4950$	$0.8 - \frac{\sqrt{m-200}}{170}$
$m > 4950$	0

- 在测试点 $9 \sim 20$ 中，设 m 表示每次调用 `perm` 函数时的询问次数的最大值，则程序将获得 $5 \cdot g(m)$ 分，其中 g 的计算方式如下表所示：

m	$g(m) =$
$m \le 30000$	1
$30000 < m \le 30015$	$1 - \frac{7(m-30000)}{5(5m-149991)}$
$30015 < m \le 60000$	$0.75 - \frac{\sqrt{m-30015}}{700}$
$m > 60000$	$0.5 - \frac{\sqrt{m-60000}}{2500}$