B. Dark Ride

题目名称	黑暗乘车
时间限制	1 秒
空间限制	1 GB

Erika 最近在波恩附近的游乐园 Phantasialand 找到了一份暑期工作。她被雇佣来控制一个黑暗乘车项目经过的房间中的灯光。

该项目会依次经过 N 个房间 (房间编号为 0 到 $N-1$)。乘车从房间 0 出发，最终到达房间 $N-1$。每个房间的灯由一个开关控制，共有 N 个开关 (编号为 0 到 $N-1$)，每个开关对应一个房间。开关 s 控制的是房间 p_s。

Erika 的上司要求她打开第一个房间和最后一个房间的灯，并关闭其余所有房间的灯。听起来很简单，对吧？她只需要打开两个开关 A 和 B，满足 $p_A = 0$ 和 $p_B = N-1$，或者 $p_B = 0$ 和 $p_A = N-1$。但不巧的是，Erika 在上司讲解控制方式时没有完全听清楚，**她已经不记得数组 p 了——也就是说，她不知道每个开关控制的是哪个房间。**

Erika 必须在她的上司察觉之前弄清楚这一点。每次游乐设施开始运行前，Erika 会先关闭所有房间的灯，然后她可以选择性地打开部分开关。随着设施依次经过各个房间，每当它从亮着的房间进入熄灭的房间，或从熄灭的房间进入亮着的房间时，Erika 都会听到乘客兴奋的尖叫声。

由于游乐设施的速度可能变化，Erika 无法直接判断哪些房间是亮着的，但她至少可以知道尖叫声的次数。也就是说，她可以得知在一次行程中，灯光状态发生了多少次变化（从亮到暗或从暗到亮）。

你能在她的上司发现之前，帮助 Erika 找出是哪两个开关控制的是第一个房间和最后一个房间的灯吗？你最多可以进行 30 次游乐设施运行实验。

交互说明

本题为交互题。

- 你的程序应首先读取一个整数 N，表示黑暗游乐设施中房间的数量。
- 接下来，你的程序需要与评测器进行交互。每次想要开始一次乘坐实验时，你需要输出一行，以问号 ? 开头，后接一个长度为 N 的仅包含字符 0（对应房间熄灯） 和 1（对应房间亮灯） 的字符串，

表示你设置的每个开关的开/关状态。随后，你的程序应读取一个整数 $\ell (0 \leq \ell < N)$表示 Erika 听到乘客尖叫的次数。

- 当你想要提交答案时，输出一行，以感叹号 ! 开头，后接两个整数 A 与 B $(0 \leq A, B < N)$ 表示你认为控制起点房间和终点房间的两个开关编号（顺序不限）。输出答案后，程序应立即退出。

评测器是非自适应的，也就是说，隐藏的数组 p 会在交互开始前就固定好。

注意：每次发起乘坐请求后，必须刷新标准输出缓冲区，否则你的程序可能会因为超时（TLE）被判为不通过。在 Python 中，如果使用 input() 读取输入，标准输出会自动刷新。在 C++ 中，可以使用 cout << endl; 来打印并刷新，若使用 printf，请搭配 fflush(stdout); 一起使用。

约束条件与评分

- $3 \leq N \leq 30\,000$.
- 你最多可以进行 30 次乘坐实验（输出最终答案不计入乘坐次数）。如果超过次数限制，将被判定为 Wrong Answer（答案错误）。

你的解法将会在若干个数据组上进行评测，每个数据组对应一定的分值。 每个数据组包含若干个测试点，只有在该组的所有测试点均通过时，才能获得该组对应的分数。

数据组	分数	额外的约束条件
1	9	$N = 3$
2	15	$N \leq 30$
3	17	$p_0 = 0$, 即开关 0 控制房间 0
4	16	N 为偶数，且两个端点房间所对应的开关分别位于数组的前半部分$(0 \leq a < \frac{N}{2})$ 和后半部分 $(\frac{N}{2} \leq b < N)$
5	14	$N \leq 1000$
6	29	没有额外的约束条件

测试工具

为了便于你调试程序，我们提供了一个简单的测试工具，你可以在 Kattis 题目页面底部的“attachments” 部分下载。该工具为可选使用，注意：正式的 Kattis 评测器与该测试工具实现方式可能不同。

使用方法如下：

创建一个输入文件，例如 sample1.in，文件的第一行应为一个整数 N，第二行为隐藏排列 $p_0, p_1, ..., p_{N-1}$，用于指定每个开关所控制的房间。例如：

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

对于 Python 程序（如 `solution.py`，通常使用 `pypy3 solution.py` 运行），可以使用以下命令运行测试工具：

```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

对于 C++ 程序，首先编译（例如使用以下命令）：

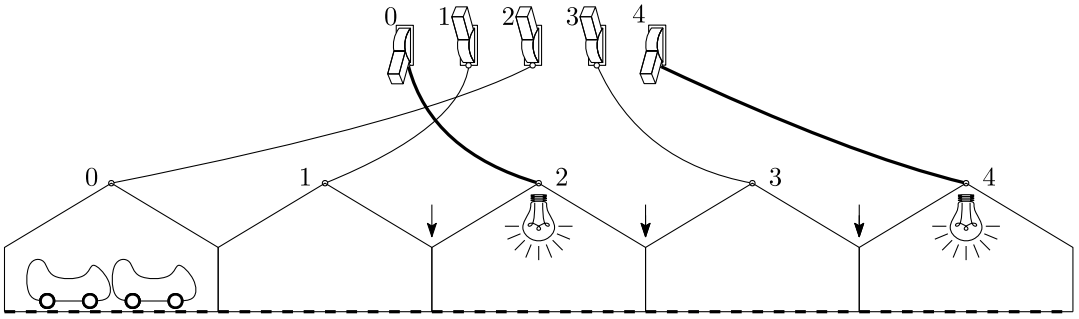
```
g++ -g -O2 -std=gnu++23 -static solution.cpp -o solution.out
```

然后运行测试工具：

```
python3 testing_tool.py ./solution.out < sample1.in
```

样例

在第一个样例中，隐藏的排列为 $[p_0, p_1, p_2, p_3, p_4] = [2, 1, 0, 3, 4]$。该排列满足数据组 2、5 和 6 的约束。首先，程序读取整数 $N = 5$。接着，程序发起一次请求，打开了 $K = 2$ 个开关，分别是开关 4 和开关 0。它们分别控制房间 $p_4 = 4$ 和 $p_0 = 2$（见下图所示）。Erika 听到了 3 声尖叫（在图中用箭头标出）：第一次是游乐设施从熄灭的房间 1 进入亮着的房间 2；第二次是从亮着的房间 2 进入熄灭的房间 3；第三次是从熄灭的房间 3 进入亮着的房间 4。随后，程序又发起一次请求，点亮了房间 p_0, p_2 和 p_3，此时 Erika 依然听到了 3 声尖叫。最后，程序提交答案 $A = 2$ 和 $B = 4$，表示这两个开关控制的是第一个和最后一个房间（即 $p_2 = 0$ 和 $p_4 = 4$），而这是一个正解。需要注意的是，输出 $A = 4$ 和 $B = 2$ 也是一个合法的答案。



在第二个样例中，隐藏的排列为 $[p_0, p_1, p_2] = [2, 0, 1]$。该排列满足数据组1、2、5 和 6 的约束。首先，程序请求打开全部三个开关，因此所有房间均被点亮，所以Erika 没有听到任何尖叫声。第二次请求时，打开了开关 1 和 0，对应点亮了房间 $p_1 = 0$ 和 $p_0 = 2$，而房间 1 没有被点亮。因此，Erika 听到了两声尖叫：第一次是亮着的房间 0 进入熄灭的房间 1；第二次是熄灭的房间 1 进入亮着的房间 2。第三次请求时，所有开关均未打开，即所有房间均未点亮，Erika 同样没有听到任何尖叫声。最后，程序输出开关 1 和 0，确实分别控制了第一个房间和最后一个房间。输出 “! 0 1” 和 “! 1 0” 均为合法答案。

在第三个样例中，隐藏的排列为 $[p_0, p_1, p_2, p_3] = [0, 1, 2, 3]$。该排列满足数据组2、3、4 、5和 6 的约束。

第一个样例

评测器输出	你的输出
5	
	? 10001
3	
	? 10110
3	
	! 2 4

第二个样例

评测器输出	你的输出
3	
	? 111
0	
	? 110
2	
	? 000
0	
	! 1 0

第三个样例

评测器输出	你的输出
4	
	? 1010
3	
	! 0 3