题目背景

在人类智慧的山巅，有着一台字长为 1048576 位的超级计算机，著名理论计算机科学家 P 博士正用它进行各种研究。不幸的是，这天台风切断了电力系统，超级计算机无法工作，而 P 博士明天就要交实验结果了，只好求助于学过 OI 的你...

题目描述

这是一道交互题。

P 博士将他的计算任务抽象为对一个整数的操作。具体来说，有一个整数 x，一开始为 0。

P 博士研究的目标是破解一种加密机器，其可以对 x 进行操作。具体来说，这个机器内部有 n 个非负整数 $p_0, p_1, \ldots, p_{n-1}$，且 $\{p_i\}$ 是 $\{0, 1, \ldots, n-1\}$ 的一个**排列**。

这个机器上有 n 个按键，编号分别为 $0, 1, \ldots, n-1$，操作第 i 个按键会使 x 加上 2^{p_i}，之后机器会输出 $popcount(x)$，即 x 的二进制表示中 1 的个数。

你需要帮助 P 博士还原机器中的排列 p。

特别的是，排列 p 是在运行前确定的，不会随选手的询问动态构造。

实现方式

你需要引用头文件 `integer.h`。

你不需要，也不应该实现主函数。你需要实现的函数接口信息如下：

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

此函数需要返回一个长度为 n 的数组，表示你还原的排列。

你可以调用的函数接口信息如下：

```
int operate(const int i);
```

此函数接受一个 $[0, n)$ 内的整数 i，表示你操作的按钮的标号。

调用此函数之后会返回一个正整数，代表操作后 x 的 $popcount$。

下发文件已经给出了一个样例选手文件 `template_integer.cpp`，可将其拷贝一份，重命名为 `integer.cpp` 并作答。

本地测试方式

下发文件中包含了两个交互库文件 `integer.h` 和 `grader.cpp`，这是我们给出的交互库参考实现。最终评测时使用的交互库实现与此略有不同，因此选手的解法不应依赖于此交互库实现。

请将你的程序命名为 `integer.cpp`，将其与交互库文件置于同一目录下，并用以下命令编译得到可执行文件 `integer`。

```
g++ grader.cpp integer.cpp -o integer -O2 -std=c++11
```

可执行文件将从标准输入读取以下格式的数据：

第一行是一个正整数 n。

第二行是一个排列 $p_0, p_1, \ldots, p_{n-1}$。

读入完成后，交互库将调用一次 `findPermutation` 并测试你的程序是否返回了正确的答案。

- 如果你的程序正确完成了操作，那么会输出 `operate` 的调用次数和你在这个测试点的得分，程序会以返回值 0 结束。
- 否则会输出第一个产生的错误信息，程序会以返回值 -1 结束。

评分标准

本题严禁以任何形式攻击交互库的行为。

如果你的程序没有正确地完成操作，那么你在这个测试点会得到 0 分。

否则，记你的程序调用 `operate` 的次数为 x，你在一个测试点得到的分数比例 s 如下：

$$s = \begin{cases} 1 & x < 8 \times 10^5 \\ \frac{8 \times 10^5}{x} & 8 \times 10^5 \le x \le 2 \times 10^6 \\ \max(\frac{180 - 20 \lg 5x}{100}, 0) & x > 2 \times 10^6 \end{cases}$$

可以发现，当交互次数超过 2×10^8 的时候，你的程序将得到 0 分。超过此次数的交互可能会导致 `TLE` 或其他不可预测的评测结果，请选手注意。

我们保证，交互库在这不超过 2×10^8 次操作中占用的时间至多为 2 秒，占用内存至多为 64 MB。

你在一个子任务中得到的分数为这个子任务的满分乘以你在这个子任务的所有测试点（包含其所依赖的所有子任务的所有测试点）得到的**最低**分数比例。

样例

我们假设输入的数据为：

```
4
0 3 1 2
```

以下是一种可能的交互过程：

$operate(0)$：$x = 1_{(2)}$，返回 1。

$operate(2)$：$x = 11_{(2)}$，返回 2。

$operate(0)$：$x = 100_{(2)}$，返回 1，可知 $p_0 + 1 = p_2$。

$operate(3)$：$x = 1000_{(2)}$，返回 1，可知 $p_2 + 1 = p_3$。

$operate(1)$：$x = 10000_{(2)}$，返回 1，可知 $p_3 + 1 = p_1$。

至此我们确定了排列 p 并退出 `findPermutation`，返回 $\{0, 3, 1, 2\}$。

子任务

对于全部数据，$1 \le n \le 5000$。

子任务 #1（5 分）：$n \le 10$。

子任务 #2（10 分）：$n \le 70$。

子任务 #3（15 分）：$n \le 900$。

子任务 #4（10 分）：$p_0 = 0$。

子任务 #5（60 分）：无特殊限制。

同时，若子任务 x 的限制**不弱于**子任务 y，有子任务 x 依赖于子任务 y。

时间限制：4s

空间限制：512MB

交互库及相关文件下载