D. 座位安排 (seatingplan)

Time limit: 4 seconds
Memory limit: 1024 MiB

這次 EGOI 的閉幕典禮將會有 N 位重要嘉賓出席。 這些嘉賓全都需要坐在第一排，並且安排的順序要按照外交禮節的種種細節。 Noemi 熬了兩個通宵來決定正確的座位順序。

Veronica 負責監督這次的閉幕典禮。她的眾多職責之一, 就是確保第一排的座位上要擺放正確的名牌。不過有個小問題：Noemi 從沒告訴過她正確的座位順序，而且現在 Noemi 人不知道在哪裏。 幸好，攝影師 Lisa 有個可能派得上用場的 App。

由於 Lisa 必須先準備她的相機，到時候才能拍到第一排嘉賓的某些特定照片。 在準備期間，她需要知道每張照片要涵蓋多寬，所以 Noemi 幫她做了一個 App，能快速輸出她需要的資訊。 Veronica 現在想用這個 App 來找出正確的座位安排。

這 N 位重要嘉賓編號為 0 到 $N-1$。 第一排的座位也從左到右編號為 0 到 $N-1$。 對於每個 I，g_I 表示應該坐在座位 I 的嘉賓編號，而 s_I 則表示嘉賓 I 應該坐的座位編號。

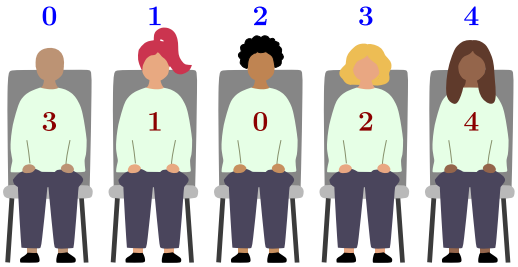

Figure 1: 這是有五位嘉賓的一排。對於這一排的表示， $g = [3, 1, 0, 2, 4]$ 且 $s = [2, 1, 3, 0, 4]$。

這款 App 的運作方式如下：

- Lisa 可輸入恰好三位不同嘉賓的編號 I、J、K。
- App 會告訴她，如果要讓這三位指定的嘉賓都在照片裡，最少會有幾位嘉賓被拍到。

以正式定義來說，App 會顯示 $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$ 這個數值。

舉例來說，請看 Figure 1 顯示的情況：

- 嘉賓 $I = 0$、$J = 2$ 及 $K = 4$ 坐在座位 $s_I = 2$、$s_J = 3$ 及 $s_K = 4$。如果 Lisa 指定他們三位的編號，App 會顯示 $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$。

換句話說，包含嘉賓 0、2 及 4 的最窄範圍，照片中剛好就只有這三位嘉賓。

- 嘉賓 $I = 0$、$J = 4$ 和 $K = 3$ 坐在座位 $s_I = 2$、$s_J = 4$ 及 $s_K = 0$。如果 Lisa 指定他們三位，App 會顯示 $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$。

換句話說，包含這三位指定嘉賓的照片，一定會拍到全部 5 位嘉賓。

請幫助 Veronica 使用 Lisa 的 App 找出正確的座位順序。 更具體來說，你的程式應該要找出並輸出數列 $g_0, g_1, \ldots, g_{N-1}$。 題目保證都恰好有兩個正確答案（兩個互相是相反順序），你可以輸出其中任一個。 你的分數會取決於你的程式方案（solution）對 App 進行的查詢次數。

Implementation

> ⇨ 這是一道互動題。你的程式需要使用標準輸入和輸出來與評測程式 (grader) 進行溝通，格式如以下所述。

你的程式一開始應該讀入一行包含一個正整數 T 的輸入，表示接下來的測試筆數。

對於每筆測資，你的程式一開始應該讀入一行包含一個正整數 N 的輸入，表示座位的數量，也表示嘉賓的數量。

要進行查詢時，你的程式應該輸出一行格式為 "? I J K" 的字串，其中 $0 \le I, J, K \le N-1$ 是三個**不同**的數字。

進行查詢後，你的程式應該讀入一行包含一個正整數的輸入，也就是查詢得到的答案。

當你要回答正確的座位順序時，你的程式應該輸出一行格式為 "! g_0 … g_{N-1}" 的字串。

在解出所有 T 筆測資後，你的程式應該要正常結束。

請注意，CMS 上用來測試你程式方案（solution）的官方評測程式可能是**自適應的 (adaptive)**。 意思是說，對於某些測資，嘉賓的排列順序並不是預先決定好的。 換言之，評測程式（grader）決定要使用哪個其餘的排列，可能取決於你的程式已詢問過的查詢。

清空輸出緩衝區 (Flushing)。如果你沒有使用所提供的模板 (templates)，請確保在印出每一行之後清空標準輸出緩衝區，否則你的程式可能會被判定為 *Not correct*。在 Python 中，如果你用 `input()` 來讀取行，輸出緩衝區會自動清空。在 C++ 中，`cout << endl;` 除了會印出換行以外，也會清空緩衝區；如果要用 printf，請使用 `fflush(stdout)`。

Constraints

- $1 \le T \le 10$。
- N 會是 5 （僅限範例）、8、40 或 2000。
- 對於每筆測資，你最多只能進行 10 000 次查詢。

Scoring

你的程式將會在分組成子任務 (subtasks) 的一些測資上進行測試。要獲得一個子任務的分數，你必須正確解出該子任務中包含的所有測資。

- **Subtask 0** [**0 points**]: 範例測資 ($N = 5$)。
- **Subtask 1** [**9 points**]: $N = 8$。
- **Subtask 2** [**11 points**]: $N = 2000$，且嘉賓 0 和 1 坐在彼此旁邊。
- **Subtask 3** [**15 points**]: $N = 40$。
- **Subtask 4** [**65 points**]: $N = 2000$。

對於子任務 1 和 2，任何能正確解出所有測資的程式方案（solution）都將獲得滿分。

對於子任務 3 和 4，你的程式方案（solution）必須正確解出所有測資才能獲得分數，而且你的分數將取決於 Q_s， 也就是你的程式方案（solution）在單一測資中所進行的最大查詢次數。令 $X_s = \max(1, Q_s/N)$。則子任務 3 和 4 的分數計算方式如下：

$$\text{score}_3 = \min\left(15, 3 + \frac{19}{X_s^{1.5}}\right), \qquad \text{score}_4 = \min\left(65, 3 + \frac{91}{X_s^{1.5}}\right)$$

每個子任務的 score_s 值會四捨五入到最接近的整數，而你的總分就是這些分數的總和。 為了拿到滿分，你需要用最多 55 次查詢解出子任務 3，以及最多 2597 次查詢解出子任務 4。下面顯示了子任務 3 和 4 的 Q_s 值和分數範例。

Q_s	55	56	60	70	80	100	150	10000
score_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
score_4	65	58	53	35	26	21	14	11

Examples

Grader	Solution
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Explanation

這筆範例輸入包含一筆測資 ($T = 1$)，其中有 $N = 5$ 位嘉賓。這筆測資裏隱藏的嘉賓座位配置對應到 Figure 1。

範例解法的第一個查詢是 0, 2, 4。 這個查詢得到的答案 3 告訴我們，這三位嘉賓以某種未知的順序，坐在三個連續的座位上。

第二個查詢的答案 3 則告訴我們，嘉賓 3、0 和 1 也一樣坐在三個連續的座位上。

我們現在可以推斷出，嘉賓 0 一定是坐在五位最中間，而嘉賓 2 和 4 坐在他的某一邊，嘉賓 1 和 3 坐在他的另外一邊。

在第三個查詢後，我們就已經能確定嘉賓們坐的順序，一定是 $[3, 1, 0, 2, 4]$ 或是相反的順序 $[4, 2, 0, 1, 3]$ 。 我們可以輸出這兩個順序中的任一個。

CMS 中的程式碼模板和評估詳情（Code Templates and Evaluation Details in CMS）

我們強烈建議你使用提供的 C++ 和 Python 程式碼模板。這些範本會檢查與評測程式的溝通是否成功，並在溝通失敗時優雅地終止程式。

如果你不使用所提供的模板, 當你的程式方案（solution）不正確時, CMS 可能會顯示錯誤的判定結果。例如, 你可能會收到 “Execution killed by signal” 或 “Execution timed out (wall clock limit exceeded)” 而不是 “Output isn’t correct”。

我們也建議你使用測試工具（Testing Tool，請參閱下文）在本地端測試你的程式方案（solution），然後再提交。此測試工具會檢查你程式方案（solution）的輸出，並報告無效查詢的使用情況。

測試工具（Testing Tool）

為了方便你測試你的程式方案（solution），我們提供了一個可以從 CMS 上下載的簡單測試工具。 你不一定要使用這個工具。請注意，CMS 上的官方評測程式（grader）和這個測試工具是不同的。

要使用這個工具時，你需要一個輸入檔。你可以使用我們提供的範例輸入檔 `seatingplan.input0.txt`, 或是自己做一個。輸入檔的第一行中應該有測資的數量 T, 然後每筆測資會有兩行：一行是一個數字 N，另一行是 $g_0, g_1, ..., g_{N-1}$ 這些數字。

對於 Python 程式，例如檔名為 `seatingplan.py`（通常用指令 `pypy3 seatingplan.py` 來執行），請用以下指令執行測試工具：

```
python3 testing_tool.py pypy3 seatingplan.py < seatingplan.input0.txt
```

對於 C++ 程式，請先編譯你的程式方案（solution）：

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o seatingplan seatingplan.cpp
```

然後執行測試工具：

```
python3 testing_tool.py ./seatingplan < seatingplan.input0.txt
```