

## D. 座席配置計画 (seatingplan)

実行時間制限: 4 秒

メモリ制限: 1024 MiB

今年の EGOI の閉会式には  $N$  名の来賓が参加する。来賓の方々については、厳しい外交儀礼に従い、最前列に決められた順番で座らせなければならない。Noemi は正しい座席配置を決める必要があり、その作業は 2 日かかりであった。

Veronica は閉会式を総括している。彼女の業務の一つは、最前列の座席の名札を正しく配置することである。しかし、残念ながら彼女は Noemi から正しい座席配置を教えてもらっておらず、かつ Noemi の現在の場所も知らないため直接聞くこともできない。だが幸運にも、写真家の Dorka が有用かもしれないアプリを持っている。

Dorka は、最前列の来賓の写真を撮影するために、カメラを準備しなければならない。その準備のために、彼女はそれぞれの写真の幅がどれくらいであるかを知ることが必要であり、そのため写真の幅を計算するアプリを作成した。いま、Veronica はそのアプリを手掛かりに、正しい座席配置を知ろうと思っている。

$N$  名の来賓には、それぞれ 0 から  $N - 1$  までの番号が付けられており、最前列の座席にも左から順に 0 から  $N - 1$  までの番号が付けられている。各  $I$  ( $0 \leq I \leq N - 1$ ) について、 $g_I$  を座席  $I$  に座る来賓の番号とし、 $s_I$  を来賓  $I$  が座る座席の番号とする。

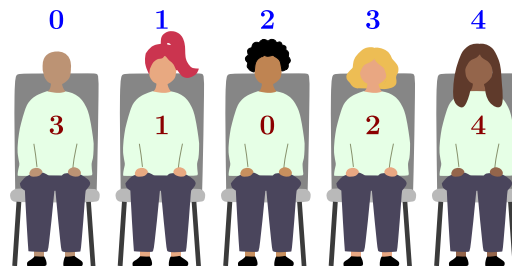


図 1: 最前列に来賓 5 名が座る場合の例。  $g = [3, 1, 0, 2, 4]$  および  $s = [2, 1, 3, 0, 4]$  となる。

このアプリは次のように動作する。

- Dorka は 3 つの相異なる来賓の番号  $I, J, K$  を入力する。
- アプリは、3 名の来賓全員が写るような写真を撮影した場合に、最小何名の来賓が写真に写ることになるかを表示する。厳密には、アプリは  $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$  の値を表示する。

例として、図 1 に示す状況を考える。

- 3 名の来賓  $I = 0, J = 2, K = 4$  はそれぞれ座席  $s_I = 2, s_J = 3, s_K = 4$  に座る。Dorka がこの入力をした場合、アプリは  $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$  を表示する。

言い換えると、来賓 0, 2, 4 が全員写るような最も狭い写真には、3 名のみが写る。

- 3 名の来賓  $I = 0, J = 4, K = 3$  はそれぞれ座席  $s_I = 2, s_J = 4, s_K = 0$  に座る。Dorka がこの入力をした場合、アプリは  $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$  を表示する。

言い換えると、3 名の来賓が全員写るような写真には、絶対に 5 名全員が写る。

Dorka のアプリを用いて正しい座席配置を求めることで、Veronica を手伝え．具体的には，あなたのプログラムは数列  $g_0, g_1, \dots, g_{N-1}$  を決定し，出力しなければならない．ここで，本問題ではちょうど 2 つの正しい解答 (ある数列と，それを逆順にした数列) が存在する．あなたはどちらを出力しても良い．あなたの得点は，アプリに何回の質問を入力したかに依存する．

## 実装方式

この問題はインタラクティブ問題である．あなたのプログラムは，採点プログラムとのやり取りを行うために，標準入力および標準出力を利用し，後述するフォーマットを使用する．

あなたのプログラムは最初に，テストケース数を表す正の整数  $T$  からなる 1 行を入力しなければならない．

各テストケースでは，正の整数  $N$  からなる 1 行を入力しなければならない．この整数は，座席数および来賓数を表す．

質問をするには，あなたのプログラムは “ $? I J K$ ” の形式で 1 行を出力しなければならない．ここで，3 つの整数は  $0 \leq I, J, K \leq N - 1$  を満たす**相異なる**整数でなければならない．

質問を行った後，あなたのプログラムは質問の回答となる 1 つの正の整数からなる 1 行を入力しなければならない．

正しい座席配置を答えるには，あなたのプログラムは “ $! g_0 \dots g_{N-1}$ ” の形式で 1 行を出力しなければならない．

$T$  個のテストケースすべてに回答した後，あなたのプログラムは正常に終了しなければならない．

なお，CMS で使用される採点プログラムのサンプルは，**適応的 (adaptive)** である．すなわち，一部のテストケースでは，来賓の座席配置を表す順列は最初から決まっているとは限らず，採点プログラムのサンプルは，あなたの質問に応じて座席配置を矛盾しないように変更する可能性がある．

**フラッシュ**．提供されたテンプレートを使用しない場合，各行の出力の後に，標準出力をフラッシュしなければならない．さもなくば，あなたのプログラムは**不正解**と判定される．Python では，入力時に `input()` を用いた場合，自動的にフラッシュが発生し，また出力時に `print(..., flush=True)` とすることで強制的にフラッシュすることができる．C++ では，改行の出力時に `cout << endl;` を用いた場合，自動的にフラッシュが発生する．printf を用いる場合，`fflush(stdout)` を使用しなければならない．

## 制約

- $1 \leq T \leq 10$ .
- $N$  は 5 (サンプルケースのみ), 8, 40, 2000 のいずれかである.
- 各テストケースについて, 最大 10000 回まで質問できる.

## 採点方式

あなたの解答は各小課題ごとに評価される. 各小課題は複数のテストケースからなる. 各小課題について得点を得るためには, その小課題に含まれるすべてのテストケースに正解する必要がある.

- **小課題 0** [ 0 点]: 入出力例 ( $N = 5$ ).
- **小課題 1** [ 9 点]:  $N = 8$ .
- **小課題 2** [11 点]:  $N = 2000$ , 来賓 0,1 は隣同士の位置に座る.
- **小課題 3** [15 点]:  $N = 40$ .
- **小課題 4** [65 点]:  $N = 2000$ .

小課題 1,2 では, 各小課題のすべてのテストケースに正解した解答は, 質問回数にかかわらず満点を得ることができる.

小課題 3,4 では, 正の得点を取るにはすべてのテストケースに正解する必要がある. また, あなたの得点は, 各小課題のテストケースにおける最大の質問回数  $Q_s$  によって決まる. 具体的には  $X_s = \max(1, Q_s/N)$  とするとき, 小課題 3,4 の得点は次式で計算される.

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

ここで,  $\text{score}_s$  は各小課題について, 最も近い整数に四捨五入され, あなたの解答の得点はすべての小課題の合計となる. 満点を獲得するには, 小課題 3 で 55 回以内, 小課題 4 で 2597 回以内の質問で解かなければならない.  $Q_s$  に対する小課題 3,4 の得点を以下に示す.

$Q_s$	55	56	60	70	80	100	150	10000
$\text{score}_3$	15	14	13	11	10	8	6	3

$Q_s$	2597	2800	3000	4000	5000	6000	8000	10000
$\text{score}_4$	65	58	53	35	26	21	14	11

## 入出力例

採点プログラム	解法
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

## 解説

サンプルテストケースでは、単一のテストケース ( $T = 1$ ) のみからなり、そのテストケースでの来賓数は  $N = 5$  である。隠された座席配置は 図 1 に対応する。

1 回目の質問は 0, 2, 4 である。質問の回答は 3 である。それは、まだ知らない座席配置において、来賓 0, 2, 4 は 3 つの連続する座席に座ることを意味する。

2 回目の質問では、来賓 3, 0, 1 についても同様であることを意味する。

その時点で、来賓 0 は中央の座席に座り、来賓 2, 4 は中央から見て片方の側、来賓 1, 3 はもう一方の側に座ることが判明する。

3 回目の質問の後、来賓が [3, 1, 0, 2, 4] またはその逆順である [4, 2, 0, 1, 3] で座る以外、あり得ないことが判明する。あなたは解答として、そのどちらを出力しても良い。

## コードテンプレートおよび CMS における採点の詳細

実装の際は、提供された C++ または Python のテンプレートを使用することを強く推奨する。これらのテンプレートは、grader とのやり取りが成功したかどうかを判定し、失敗した場合に実行を正常終了する。

あなたのプログラムとやり取りをしている採点プログラムのサンプルは、遭遇した最初のエラーを報告し、終了する。もし提供されたテンプレートを使用しなかった場合、あなたのプログラムがクラッシュしたり、質問への回答を待ち続けたりする場合がある。その場合、CMS の採点結果のメッセージはあまり参考にならず、たとえば “Execution timed out (wall clock limit exceeded)” 等になるかもしれない。

提出の前に、テストのためのツール (後述) を使用し、ローカル環境でテストを行うことを強く推奨する。本ツールはプログラムの出力を検証し、どのプロトコルに違反したのかを報告する。

## テストのためのツール

あなたが解法をテストすることを容易にするため、簡単なツールが CMS からダウンロードできるようになっている。ここで、本ツールを必ずしも使う必要はない。なお、CMS で使用される実際の採点プログラムは、本ツールとは異なる。

本ツールを使うには、まず入力ファイルを指定しなければならない。あなたは、提供されたサンプルテストケースの入力 `seatingplan.input0.txt` を使用しても良く、また独自のテストケースを使用しても良い。入力ファイルはテストケース数  $T$  からなる行から始まり、その後各テストケースにつき 2 行で構成される。1 行目は整数  $N$  であり、2 行目は整数  $g_0, g_1, \dots, g_{N-1}$  である。

Python の場合、たとえば `seatingplan.py` の場合 (通常 `pypy3 seatingplan.py` とすれば実行されるが)、以下を実行しなければならない。

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

C++ の場合、まずコンパイルを行わなければならない。

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

続いて、以下のような形式で、テストのためのツールを実行しなければならない。

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