

## D. План розсадки (seatingplan)

Церемонію закриття цієї EGOI відвідають  $N$  важливих гостей. Усі вони мають сидіти в першому ряду в дуже специфічному порядку, який відповідає всім нюансам дипломатичного протоколу. Визначення правильного порядку розсадки зайняло в Ноемі дві безсонні ночі.

Вероніка керує церемонією закриття. Один із її багатьох обов'язків — переконатися, що на місцях у першому ряду є правильні таблички з іменами. Є лише одна невелика проблема: Ноемі так і не сказала їй правильний порядок розсадки, а тепер Ноемі ніде немає. На щастя, фотографка Дорка має застосунок, який може бути корисним.

Дорка мала підготувати свої камери так, щоб зробити кілька конкретних фотографій гостей у першому ряду. Для налаштування їй потрібно було знати, якої ширини буде кожне фото, тому Ноемі створила для неї застосунок, який швидко видає потрібну інформацію. Тепер Вероніка хоче використати цей застосунок, щоб знайти правильний порядок розсадки.

$N$  важливих гостей пронумеровані від 0 до  $N - 1$ . Місця в першому ряду також пронумеровані від 0 до  $N - 1$ , зліва направо. Для кожного  $I$  ( $0 \leq I \leq N - 1$ ) нехай  $g_I$  позначає гостя, який має сидіти на місці  $I$ , а  $s_I$  позначає місце, на якому має сидіти гість  $I$ .

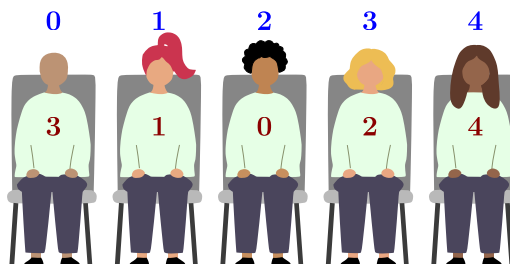


Рисунок 1: Ряд із п'ятьма гостями. Для цього ряду  $g = [3, 1, 0, 2, 4]$  та  $s = [2, 1, 3, 0, 4]$ .

Застосунок працює наступним чином:

- Дорка вводить номери  $I$ ,  $J$ ,  $K$  рівно трьох різних гостей.
- Застосунок повідомляє їй мінімальну кількість гостей, яких буде видно, якщо всі троє вибраних гостей будуть на фотографії. Формально, застосунок поверне значення  $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$ .

Наприклад, розглянемо ситуацію, показану на Рисунок 1:

- Гості  $I = 0$ ,  $J = 2$  та  $K = 4$  сидять на місцях  $s_I = 2$ ,  $s_J = 3$  та  $s_K = 4$ . Якщо Дорка обере їх, застосунок відобразить значення  $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$ .

Іншими словами, найвужче фото, на якому є гості 0, 2 та 4, містить лише цих трьох гостей.

- Гості  $I = 0$ ,  $J = 4$  та  $K = 3$  сидять на місцях  $s_I = 2$ ,  $s_J = 4$  та  $s_K = 0$ . Якщо Дорка обере їх, застосунок відобразить значення  $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$ .

Іншими словами, фотографія, яка містить трьох заданих гостей, повинна містити всіх 5 гостей.

Допоможіть Вероніці визначити правильний порядок розсадки за допомогою застосунку Дорки. Більш конкретно, ваша програма повинна визначити та вивести послідовність  $g_0, g_1, \dots, g_{N-1}$ . Завжди існують рівно дві правильні відповіді (одна є зворотною до іншої), і ви можете вивести

будь-яку з них. Ваша оцінка залежатиме від кількості запитів до застосунку, зроблених вашим розв'язком.

## Реалізація



Це інтерактивна задача. Ваша програма використовуватиме стандартні потоки введення та виведення для обміну даними з грейдером у форматі, описаному нижче.

Ваша програма повинна розпочати з читання одного рядка вхідних даних, що містить додатне ціле число  $T$  — кількість тестових наборів, які йдуть далі.

Для кожного тестового набору ваша програма повинна розпочати з читання одного рядка вхідних даних, що містить додатне ціле число  $N$  — кількість місць, яка також є кількістю гостей.

Щоб зробити запит, ваша програма повинна вивести рядок у форматі «?  $I$   $J$   $K$ », де  $0 \leq I, J, K \leq N - 1$  — три **різні** числа.

Після виконання запиту ваша програма повинна прочитати один рядок, що містить одне додатне ціле число — відповідь на ваш запит.

Щоб надати правильний порядок розсадки у якості відповіді, ваша програма повинна вивести рядок у форматі «!  $g_0 \dots g_{N-1}$ ».

Після розв'язання всіх  $T$  тестових наборів ваша програма повинна штатно завершити роботу.

Зверніть увагу, що офіційний грейдер, який використовується в CMS для тестування вашого розв'язку, може бути **адаптивним**. Це означає, що для деяких тестових наборів перестановка гостей не визначена заздалегідь. Натомість грейдер може вирішити, яку з решти перестановок використовувати, залежно від запитів, які робить ваша програма.

**Скидання буфера (Flushing).** Якщо ви не використовуєте надані шаблони, обов'язково скидайте буфер стандартного виведення після друку кожного рядка, інакше ваша програма може отримати вердикт *Неправильна відповідь (Not correct)*. У Python це відбувається автоматично, якщо ви використовуєте `input()` для читання рядків, і ви можете використовувати `print(..., flush=True)`, щоб примусово скинути буфер. У C++ `cout << endl`; скидає буфер на додаток до друку символу нового рядка; якщо ви використовуєте `printf`, застосовуйте `fflush(stdout)`.

## Обмеження

- $1 \leq T \leq 10$ .
- $N$  дорівнюватиме 5 (лише у прикладі), 8, 40 або 2000.
- Для кожного тестового набору ви можете зробити не більше ніж 10 000 запитів.

## Оцінювання

Ваша програма буде протестована на кількох тестових наборах, згрупованих у підзадачі. Щоб отримати бали за підзадачу, ви повинні правильно розв'язати всі тести, які вона містить.

- **Підзадача 0** [ 0 балів]: Приклад ( $N = 5$ ).
- **Підзадача 1** [ 9 балів]:  $N = 8$ .
- **Підзадача 2** [11 балів]:  $N = 2000$ , і гості 0 та 1 сидять поруч.
- **Підзадача 3** [15 балів]:  $N = 40$ .
- **Підзадача 4** [65 балів]:  $N = 2000$ .

Для підзадач 1 і 2 будь-який розв'язок, який правильно розв'язує всі тестові набори, отримає всі бали.

Для підзадач 3 і 4 ваш розв'язок повинен правильно розв'язати всі тестові набори, щоб отримати будь-які бали, і ваша оцінка залежатиме від  $Q_s$  — найбільшої кількості запитів, які знадобилися

вашому розв'язку для розв'язання одного тестового набору. Нехай  $X_s = \max(1, Q_s/N)$ . Бали за підзадачі 3 і 4 обчислюються наступним чином:

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

Значення  $\text{score}_s$  округлюється до найближчого цілого числа для кожної підзадачі, а ваша загальна оцінка є їхньою сумою. Щоб отримати повний бал, вам потрібно розв'язати підзадачу 3, використавши не більше 55 запитів, і підзадачу 4, використавши не більше 2597 запитів. Приклади значень  $Q_s$  та балів для підзадач 3 і 4 наведені нижче.

$Q_s$	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

### Приклади вводу/виводу

Грейдер	Розв'язок
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

### Пояснення

Приклад вхідних даних містить один тестовий набір ( $T = 1$ ) із  $N = 5$  гостями. Прихована конфігурація гостей у цьому тестовому наборі відповідає Рисунку 1.

Перший запит, зроблений у прикладі розв'язку, це 0, 2, 4. Відповідь 3 на цей запит говорить нам, що ці гості сидять у якомусь невідомому порядку на трьох послідовних місцях поруч один з одним.

Відповідь 3 на другий запит говорить нам те саме про гостей 3, 0 та 1.

Тепер ми можемо зробити висновок, що гість 0 має сидіти посередині, причому гості 2 і 4 — з одного боку, а гості 1 і 3 — з іншого.

Після третього запиту ми вже можемо бути впевнені, що гості повинні сидіти або в порядку [3, 1, 0, 2, 4], або у зворотному порядку [4, 2, 0, 1, 3]. Можна вивести будь-який із цих порядків.

### Шаблони коду та деталі оцінювання в CMS

Ми наполегливо рекомендуємо використовувати надані шаблони коду для C++ та Python. Вони перевіряють, чи була комунікація з грейдером успішною, і штатно завершують роботу, якщо це не так.

Якщо ви не використовуєте надані шаблони, у випадках, коли ваше рішення неправильне, CMS може відобразити неправильний вердикт. Наприклад, замість вердикту «Output isn't correct» ви можете отримати «Execution killed by signal» або «Execution timed out (wall clock limit exceeded)»

Ми також рекомендуємо використовувати інструмент тестування (див. нижче) для локальної перевірки вашого розв'язку перед його відправленням. Інструмент тестування перевіряє вивід вашого розв'язку та повідомляє про порушення протоколу.

## Інструмент тестування

Для полегшення тестування вашого розв'язку ми надаємо простий інструмент, який ви можете завантажити з CMS. Використання інструменту не є обов'язковим. Зверніть увагу, що офіційний грейдер у CMS відрізняється від інструменту тестування.

Щоб скористатися інструментом, вам потрібен файл із вхідними даними. Ви можете використати наданий приклад вхідних даних `seatingplan.input0.txt` або створити свій власний. Файл із вхідними даними повинен починатися з рядка, який містить кількість тестових наборів  $T$ , а потім він повинен містити по два рядки на кожний тестовий набір: один рядок із числом  $N$  і один рядок із числами  $g_0, g_1, \dots, g_{N-1}$ .

Для програм на Python, наприклад `seatingplan.py` (які зазвичай запускаються як `pyru3 seatingplan.py`), запустіть інструмент тестування наступним чином:

```
python3 testing_tool.py pyru3 seatingplan.py < seatingplan.input0.txt
```

Для програм на C++ спочатку скомпілюйте ваш розв'язок:

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

а потім запустіть інструмент тестування:

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