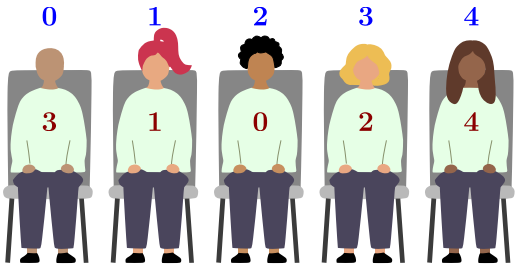
D. مخطط الجلوس (seatingplan)

سيحضر حفل ختام EGOI هذا N من الضيوف المهمين. يحتاج هؤلاء جميعاً للجلوس في الصف الأمامي بترتيب دقيق جداً يوافق كل تفاصيل البروتوكول الدبلوماسي. استغرقت عملية تحديد ترتيب الجلوس الصحيح ليلتين بلا نوم من «نومي».

«فيرونيكا» هي المشرفة على حفل الختام. من بين مسؤولياتها الكثيرة التأكد من أن مقاعد الصف الأمامي تحمل لافتات الأسماء الصحيحة. هناك مشكلة صغيرة واحدة فقط: «نومي» لم تخبرها أبداً بترتيب الجلوس الصحيح، والآن لا أثر لـ «نومي». لحسن الحظ، لدى «دوركا» المصورة تطبيق قد يكون مفيداً.

كان على «دوركا» تجهيز كاميراتها لتتمكن من التقاط صور معينة للضيوف في الصف الأول. للإعداد، كانت تحتاج لمعرفة مدى اتساع كل صورة، لذا صنعت لها «نومي» تطبيقاً يعطيها المعلومات التي تحتاجها بسرعة. تريد «فيرونيكا» الآن استخدام التطبيق لمعرفة توزيع المقاعد الصحيح.

الضيوف الـ N المهمون مرقمون من 0 إلى $N-1$. مقاعد الصف الأمامي مرقمة أيضاً من 0 إلى $N-1$، من اليسار إلى اليمين. لكل I $(0 \leq I \leq N-1)$، دعنا نرمز بـ g_I للضيف المفترض أن يجلس في المقعد I، وبـ s_I للمقعد الذي يُفترض أن يجلس فيه الضيف I.



شكل 1: صف به خمسة ضيوف. لهذا الصف، $g = [3, 1, 0, 2, 4]$ و $s = [2, 1, 3, 0, 4]$.

يعمل التطبيق كالتالي:

- تدخل «دوركا» أرقام I و J و K لثلاثة ضيوف مختلفين تماماً.
- يخبرها التطبيق بالحد الأدنى لعدد الضيوف الذين سيظهرون إذا كان هؤلاء الضيوف الثلاثة المختارون في الصورة. رسمياً، سيعرض التطبيق القيمة $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

على سبيل المثال، انظر للحالة الموضحة في شكل 1:

- الضيوف $I = 0$ و $J = 2$ و $K = 4$ يجلسون في المقاعد $s_I = 2$ و $s_J = 3$ و $s_K = 4$. إذا اختارتهم «دوركا»، سيعرض التطبيق القيمة $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  بمعنى آخر، أضيق صورة تحتوي على الضيوف 0 و 2 و 4 تضم هؤلاء الضيوف الثلاثة فقط.

- الضيوف $I = 0$ و $J = 4$ و $K = 3$ يجلسون في المقاعد $s_I = 2$ و $s_J = 4$ و $s_K = 0$. إذا اختارتهم «دوركا»، سيعرض التطبيق القيمة $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  بمعنى آخر، أي صورة تحتوي على الضيوف الثلاثة المحددين يجب أن تحتوي على جميع الضيوف الـ 5.

ساعد «فيرونيكا» في تحديد ترتيب الجلوس الصحيح باستخدام تطبيق «دوركا». بتعبير أدق، يجب على برنامجك تحديد وإخراج المتسلسلة $g_0, g_1, \ldots, g_{N-1}$. هناك دائماً إجابتان صحيحتان بالضبط (إحداهما هي عكس الأخرى)، ويمكنك إخراج أي منهما. ستعتمد درجتك على عدد الاستعلامات التي يطلبها حلك من التطبيق.

التنفيذ

⇨ هذه مسألة تفاعلية. سيستخدم برنامجك المدخلات والمخرجات القياسية للتواصل مع نظام التقييم (Grader) بالصيغة الموضحة أدناه.

يجب أن يبدأ برنامجك بقراءة سطر واحد من المدخلات يحتوي على عدد صحيح موجب T، وهو عدد حالات الاختبار التي ستلي.

لكل حالة اختبار، يجب أن يبدأ برنامجك بقراءة سطر واحد من المدخلات يحتوي على عدد صحيح موجب N، وهو عدد المقاعد، والذي يمثل أيضاً عدد الضيوف.

لإجراء استعلام، يجب أن يخرج برنامجك سطراً بصيغة «? $K\ J\ I$»، حيث $0 \le I, J, K \le N-1$ هي ثلاثة أرقام **مختلفة**.

بعد إجراء الاستعلام، يجب أن يقرأ برنامجك سطراً واحداً يحتوي على عدد صحيح موجب، وهو إجابة استعلامك.

للإجابة بترتيب جلوس صحيح، يجب أن يخرج برنامجك سطراً بصيغة «! g_0 ... g_{N-1}».

بعد حل جميع حالات الاختبار الـ T، يجب أن ينهي برنامجك عمله بشكل طبيعي.

لاحظ أن نظام التقييم الرسمي المستخدم في CMS لاختبار حلك قد يكون **تفاعلياً**. أي أنه في بعض حالات الاختبار، لا يتم تحديد ترتيب الضيوف مسبقاً. بدلاً من ذلك، قد يقرر نظام التقييم أي الترتيبات المتبقية سيستخدمها بناءً على الاستعلامات التي طرحها برنامجك بالفعل.

تفريغ المخرجات (Flushing). إذا كنت لا تستخدم القوالب الموفرة، تأكد من تفريغ المخرجات القياسية (flush) بعد طباعة كل سطر، وإلا فقد يتم الحكم على برنامجك بأنه غير صحيح. في لغة بايثون، يحدث هذا تلقائياً إذا كنت تستخدم `input()` لقراءة الأسطر، ويمكنك استخدام `print(..., flush=True)` لإجبارها على التفريغ. في لغة C++، تقوم `cout << endl;` بالتفريغ بالإضافة إلى طباعة سطر جديد؛ إذا كنت تستخدم printf، استخدم `fflush(stdout)`.

القيود

- $1 \le T \le 10$.
- N ستكون 5 (مثال فقط)، 8، 40، أو 2000.
- لكل حالة اختبار، يمكنك إجراء ما لا يزيد عن 10 000 استعلام.

توزيع الدرجات

سيتم اختبار برنامجك على عدة حالات اختبار مجمعة في مهام فرعية. للحصول على درجة مهمة فرعية، يجب أن تحل جميع الاختبارات التي تحتويها بشكل صحيح.

- **المهمة الفرعية 0 [0 نقاط]**: مثال ($N = 5$).
- **المهمة الفرعية 1 [9 نقاط]**: $N = 8$.
- **المهمة الفرعية 2 [11 نقاط]**: $N = 2000$، والضيفان 0 و 1 يجلسان بجانب بعضهما.
- **المهمة الفرعية 3 [15 نقاط]**: $N = 40$.
- **المهمة الفرعية 4 [65 نقاط]**: $N = 2000$.

بالنسبة للمهام الفرعية 1 و 2، أي حل يحل جميع حالات الاختبار بشكل صحيح سيحصل على كامل النقاط.

بالنسبة للمهام الفرعية 3 و 4، يجب أن يحل حلك جميع حالات الاختبار بشكل صحيح ليحصل على أي نقاط، و ستعتمد درجتك على Q_s، وهو أكبر عدد من الاستعلامات التي احتاج حلك لإجرائها لحل حالة اختبار واحدة. لنفترض أن $X_s = \max(1, Q_s/N)$. يتم حساب درجات المهام الفرعية 3 و 4 كما يلي:

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

يتم تقريب قيمة score_s لأقرب عدد صحيح لكل مهمة فرعية، ودرجتك الإجمالية هي مجموع هذه الدرجات. للحصول على الدرجة الكاملة، تحتاج إلى حل المهمة الفرعية 3 بحد أقصى 55 استعلاماً والمهمة الفرعية 4 بحد أقصى 2597 استعلاماً. أمثلة لقيم Q_s والدرجات للمهام الفرعية 3 و 4 موضحة أدناه.

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

أمثلة للإدخال/الإخراج

الحل	مقيّم
	1 5
? 0 2 4	
	3
? 3 0 1	
	3
? 0 4 3	
	5
! 3 1 0 2 4	

الشرح

تحتوي مدخلات المثال على حالة اختبار واحدة ($T = 1$) بها $N = 5$ ضيوف. توزيع الضيوف الخفي في حالة الاختبار هذه يتوافق مع شكل 1.

الاستعلام الأول الذي أجراه حل المثال هو 0، 2، 4. إجابة هذا الاستعلام وهي 3 تخبرنا أن هؤلاء الضيوف يجلسون، بترتيب غير معروف، في ثلاثة مقاعد متتالية بجانب بعضهم البعض.

إجابة الاستعلام الثاني وهي 3 تخبرنا بنفس الشيء عن الضيوف 3 و 0 و 1.

يمكننا الآن استنتاج أن الضيف 0 يجب أن يجلس في المنتصف، مع وجود الضيوف 2 و 4 على جانب والضيوف 1 و 3 على الجانب الآخر.

بعد الاستعلام الثالث، يمكننا بالفعل التأكد من أن الضيوف يجب أن يجلسوا إما بالترتيب $[3, 1, 0, 2, 4]$ أو بالترتيب المعكوس $[4, 2, 0, 1, 3]$. يمكننا إخراج أي من هذين الترتيبين.

قوالب الكود وتفاصيل التقييم في CMS

ننصح بشدة باستخدام قوالب الكود الموفرة للغتي C++ وبايثون. فهي تتحقق مما إذا كان التواصل مع نظام التقييم ناجحاً وتنهي البرنامج بسلاسة في حال لم يكن كذلك.

إذا لم تستخدم القوالب الموفرة، في الحالات التي يكون فيها حلك غير صحيح، قد يعرض نظام CMS حكماً خاطئاً. على سبيل المثال، بدلاً من «Output isn't correct» قد تحصل على «Execution killed by signal» أو «Execution timed out (wall clock limit exceeded)».

ننصح أيضاً باستخدام أداة الاختبار (انظر أدناه) لاختبار حلك محلياً قبل تسليمه. تتحقق أداة الاختبار من مخرجات حلك وتُبلغ عن استخدام استعلامات غير صالحة.

أداة الاختبار

لتسهيل اختبار حلك، نوفر أداة بسيطة يمكنك تنزيلها من CMS. استخدام الأداة اختياري. لاحظ أن نظام التقييم الرسمي على CMS يختلف عن أداة الاختبار.

لاستخدام الأداة، تحتاج إلى ملف مدخلات. يمكنك استخدام مثال المدخلات الموفر `seatingplan.input0.txt` أو إنشاء ملف خاص بك. يجب أن يبدأ ملف المدخلات بسطر يحتوي على عدد حالات الاختبار T ثم يجب أن يحتوي على سطرين لكل حالة اختبار: سطر يحتوي على العدد N ثم سطر يحتوي على الأرقام $g_0, g_1, \ldots, g_{N-1}$.

لبرامج بايثون، لنفترض أن اسم الملف `seatingplan.py` (يُشغل عادةً كـ `pypy3 seatingplan.py`) قم بتشغيل أداة الاختبار كما يلي:

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

لبرامج C++، قم أولاً بتجميع حلك:

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

ثم قم بتشغيل أداة الاختبار:

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