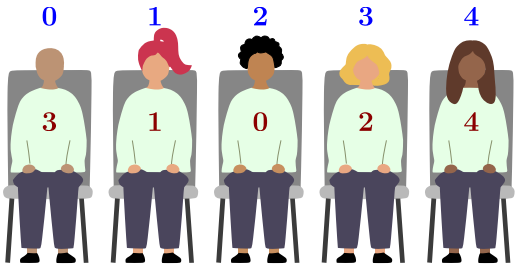
D. خطة الجلوس (seatingplan)

سيحضر حفل ختام EGOI هذا N من الضيوف المهمين. يجب جلوسهم جميعًا في الصف الأول بترتيب محدد جدًا يتوافق مع كل تفاصيل البروتوكول الدبلوماسي. استغرق تحديد ترتيب الجلوس الصحيح من نويمي ليلتين بلا نوم.

فيرونيكا تشرف على حفل الختام. إحدى مسؤولياتها العديدة هي التأكد من وجود لافتات الأسماء الصحيحة على مقاعد الصف الأول. هناك مشكلة صغيرة واحدة فقط: لم تخبرها نويمي أبدًا بترتيب الجلوس الصحيح، والآن لا يمكن العثور على نويمي في أي مكان. لحسن الحظ، لدى ليزا المصورة تطبيق قد يكون مفيدًا.

كان على ليزا تجهيز كاميراتها لتتمكن من التقاط صور محددة لضيوف في الصف الأول. للإعداد، كانت بحاجة لمعرفة مدى عرض كل صورة، لذلك صنعت لها نويمي تطبيقًا يُخرج بسرعة المعلومات التي تحتاجها. تريد فيرونيكا الآن استخدام التطبيق لمعرفة توزيع المقاعد الصحيح.

الضيوف المهمون N مرقمون من 0 إلى $N-1$. المقاعد في الصف الأمامي مرقمة أيضًا من 0 إلى $N-1$ من اليسار إلى اليمين. لكل I، لندع g_I يرمز للضيف الذي من المفترض أن يجلس في المقعد I، ولندع s_I يرمز للمقعد الذي من المفترض 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 ليزا أرقام I و J و K لثلاثة ضيوف مختلفين تمامًا.
- يخبر التطبيق ليزا بالحد الأدنى لعدد الضيوف الذين سيكونون مرئيين إذا كان جميع الضيوف الثلاثة المختارين في الصورة. رسميًا، سيعرض التطبيق القيمة $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

على سبيل المثال، انظر إلى الموقف الموضح 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 ليزا، سيعرض التطبيق القيمة $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  بعبارة أخرى، 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 ليزا، سيعرض التطبيق القيمة $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  بعبارة أخرى، الصورة التي تحتوي على الضيوف الثلاثة المعطيين يجب أن تحتوي على جميع الضيوف الـ 5.

ساعد فيرونيكا في تحديد ترتيب الجلوس الصحيح باستخدام تطبيق ليزا. بشكل أكثر تحديدًا، يجب على برنامجك تحديد وإخراج التسلسل $g_0, g_1, \ldots, g_{N-1}$. توجد دائمًا إجابتان صحيحتان بالضبط (واحدة هي عكس الأخرى)، ويمكنك إخراج أي منهما. ستعتمد نتيجتك على عدد الاستعلامات التي يرسلها حلك للتطبيق.

التنفيذ

⇨ هذه مسألة تفاعلية. سيستخدم برنامجك المدخلات والمخرجات القياسية للتواصل مع المصحح بالتنسيق الموضح أدناه.

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

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

لإجراء استعلام، يجب على برنامجك إخراج سطر بالشكل «? I J K»، حيث $0 \leq I, J, K \leq N-1$ هي ثلاثة أرقام **مختلفة**.

بعد إجراء استعلام، يجب على برنامجك قراءة سطر واحد يحتوي على عدد صحيح موجب، وهو إجابة استعلامك.

للإجابة بترتيب الجلوس الصحيح، يجب على برنامجك إخراج سطر بالشكل «! g_0 ... g_{N-1}».

بعد حل جميع حالات الاختبار T، يجب أن ينتهي برنامجك بشكل طبيعي.

لاحظ أن المصحح الرسمي المستخدم في CMS لاختبار حلك قد يكون **متأقلما**. أي أنه بالنسبة لبعض حالات الاختبار، لا يتم تحديد تبديل الضيوف مسبقًا. بدلاً من ذلك، قد يغير المصحح سلوكه اعتمادًا على الاستعلامات المحددة التي يطرحها برنامجك.

تفريغ المخزن المؤقت (Flushing). إذا كنت لا تستخدم القوالب المقدمة، فتأكد من تفريغ المخرج القياسي بعد طباعة كل سطر، وإلا فقد يتم الحكم على برنامجك بأنه غير صحيح. في بايثون، يحدث هذا تلقائيًا إذا كنت تستخدم `input()` لقراءة الأسطر. في C++، `cout << endl;` تقوم بالتفريغ بالإضافة إلى طباعة سطر جديد؛ إذا كنت تستخدم printf، استخدم `fflush(stdout)`.

القيود

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

توزيع الدرجات

سيتم اختبار برنامجك على عدة حالات اختبار مجمعة في مهام فرعية. للحصول على درجة مهمة فرعية، يجب عليك حل جميع الاختبارات التي تحتوي عليها 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 البعض.
- **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 للحصول على أي نقاط، وستعتمد نتيجتك على Q_s، وهو أكبر عدد من الاستعلامات التي يحتاجها حلك لحل حالة اختبار. لنفترض أن $X_s = \max(1, Q_s/N)$. يتم حساب درجات المهام الفرعية 3 و 4 بعد ذلك 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 إلى أقرب عدد صحيح لكل مهمة فرعية، ونتيجتك الإجمالية هي مجموع هذه القيم. للحصول على درجة كاملة، تحتاج إلى حل المهمة الفرعية 3 في 55 استعلامًا على الأكثر والمهمة الفرعية 4 في 2597 استعلامًا على الأكثر.

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

مثال المدخلات يحتوي على حالة اختبار واحدة $(T = 1)$ مع $N = 5$ ضيوف. تكوين الضيوف الخفي في حالة الاختبار هذه يتوافق مع شكل 1.

أول استعلام قام به الحل النموذجي هو 0، 2، 4. الإجابة 3 لهذا الاستعلام تخبرنا أن هؤلاء الضيوف يجلسون، بترتيب غير معروف، في ثلاثة مقاعد متتالية بجانب بعضهم البعض.

الإجابة 3 للاستعلام الثاني 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 العكسي $[4, 2, 0, 1, 3]$. يمكننا إخراج أي من الترتيبين.

أداة الاختبار

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

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

لبرامج بايثون، لنقل `seatingplan.py` (عادة ما يتم تشغيله كـ `pypy3 seatingplan.py`) شغل 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 شغل أداة الاختبار:

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