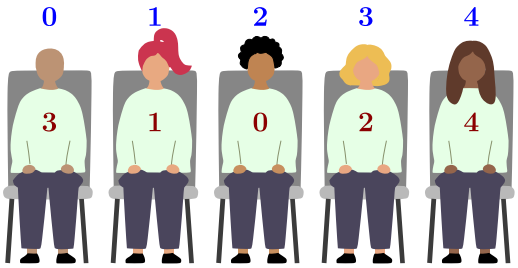
# D. Dokonalé usadenie (seatingplan)

Na záverečnej ceremónii budú účastníčky usadené podľa dopredu pripraveného plánu, ktorý je samozrejme záhadný, nepredvídeľný a nemenný. Organizátori sa poučili z otváracieho ceremoniálu a rozhodli sa usadiť ich do jedného radu[1].

Na EGOI je $N$ účastníčiek, číslovaných od 0 po $N-1$. Sedadlá v rade tiež číslujeme od 0 po $N-1$, zľava doprava. Pre každé $0 \leq I \leq N-1$, nech $g_I$ označuje číslo účastníčky ktorá by mala sedieť v sedadle číslo $I$ a nech $s_I$ označuje sedadlo, na ktorom by mala sedieť účastníčka $I$.




Obr. 1: Pre tento rad s piatimi účastníčkami platí $g = [3, 1, 0, 2, 4]$ a $s = [2, 1, 3, 0, 4]$.

Všetko by bolo dobre, keby niekto nedal tento zasadací poriadok Franciscovi a ten ho nestratil. Záverečený ceremoniál má byť už o hodinu a presne určené poradie sa predsa nemôže meniť![2]. Mišof však potrebuje stihnúť svoj vlak a tak je na ňom, aby poradie čo najrýchlejšie zrekonštruoval.

Mišof požiadal o pomoc Adri, keďže má tendenciu pamätať si náhodné údaje. A našťastie si pamätá zrovna také, ktoré sú aspoň trochu užitočné.

Konkrétne, ak sa Mišof Adri spýta na ľubovoľnú trojicu *rôznych* účastníčok $I, J, K$, potom si Adri spomenie, *koľko sedadiel (vrátane tých, na ktorých sedia) je v úseku tvorenom touto trojicou*. Formálne, Adri odpovie hodnotu $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Napríklad si pozri situáciu zobrazenú na Obr. 1:

- Dievčatá $I = 0$, $J = 2$ a $K = 4$ sedia na sedadlách $s_I = 2$, $s_J = 3$ a $s_K = 4$. Adri si pamätá, že úsek obsahujúci tieto dievčatá má $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$ sedadiel.
- Dievčatá $I = 0$, $J = 4$ a $K = 3$ sú na sedadlách $s_I = 2$, $s_J = 4$ a $s_K = 0$. Ak sa Mišof opýta na tieto tri, Adri odpovie $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

Ceremoniál (aj vlak) sa ale blíži, takže možno nie je dosť času pýtať sa na všetky trojice! Pomôžte Mišofovi zistiť, aké je poradie sedenia na záverečnom ceremoniáli, skôr, než mu ujde vlak!

Presnejšie, váš program by mal zistiť a vypísať postupnosť $g_0, g_1, ..., g_{N-1}$. Vždy existujú presne dve správne odpovede (jedna je reverznutou verziou tej druhej) a môžete vypísať hociktorú z nich. Vaše skóre bude závisieť od počtu otázok, ktoré váš program použije.

## Implementácia

⇨ Toto je interaktívna úloha. Váš program bude komunikovať s graderom pomocou štandardného vstupu a výstupu vo formáte popísanom nižšie.

[1] aby každá mala dobrý výhľad
[2] na rozdiel od začiatočného ceremoniálu

V rámci jedného vstupu musí váš program vyriešiť problém pre $T$ zatváracích ceremoniálov.

Váš program by mal začať načítaním riadku, obsahujúceho čísla $T$.

Pre každú testovaciu sadu by mal váš program najprv načítať jeden riadok s kladným celým číslom $N$ – počet účastníčiek a zároveň aj počet sedadiel.

Váš program by pre každú sadu mal následne zistiť zasadací poriadok. Na spýtanie sa otázky vypíšte riadok vo formáte „? $I$ $J$ $K$“, kde $0 \leq I, J, K \leq N-1$ sú tri **rôzne** čísla.

Po vypísaní otázky by váš program mal načítať jeden riadok obsahujúci jedno kladné celé číslo – odpoveď na otázku.

Na vypísanie správneho zasadacieho poriadku by váš program by mal vypísať riadok vo formáte „! $g_0 \cdots g_{N-1}$“.

Po vyriešení všetkých $T$ testovacích sád by sa mal váš program normálne ukončiť.

Oficiálny testovač v CMS, ktorý testuje vaše riešenie, môže byť **adaptívny**. To znamená, že v niektorých testovacích sadách nie je permutácia hostí určená vopred. Namiesto toho sa môže grader priebežne rozhodovať, ako vám odpovie, podľa toho, aké otázky ste položili doteraz a aké prípustné permutácie ešte existujú.

Na správne vyriešenie vstupu musíte správne odpovedať pre všetky zatváracie ceremoniály na vstupe.

**Flushing.** Ak nepoužívate poskytnuté šablóny, dajte si pozor, aby ste po vypísaní každého riadku flushli výstup, inak by mohol váš program dostať verdikt *Not correct* (nesprávne). V Pythone sa to deje samo, ak na načítavanie riadkov používaš `input()`, a ak potrebuješ, môžeš ho k tomu donútiť pomocou `print(..., flush=True)`. V C++ robí `cout << endl;` okrem vypísania nového riadku aj flush; ak používaš printf, použi `fflush(stdout)`.

## Obmedzenia

- $1 \leq T \leq 10$.
- $N$ bude 5 (len v príklade), 8, 40 alebo 2000.
- Pre každý zatvárací ceremionál na vstupe váš program smie použiť najviac 10 000 otázok.

## Bodovanie

Existuje viacero testovacích sád, každá s inými obmedzeniami. Body za sadu dostanete v prípade, že pre každý zatvárací ceremionál vo vstupoch váš program odpovie správne, *s použitím najviac* 10 000 *otázok. Presný počet bodov záleží od maximálneho počtu použitých otázok.*

- **Podúloha 0** [ **0 bodov**]: Príklad ($N = 5$).
- **Podúloha 1** [ **9 bodov**]: $N = 8$.
- **Podúloha 2** [**11 bodov**]: $N = 2000$ a hostia 0 a 1 sedia vedľa seba.
- **Podúloha 3** [**15 bodov**]: $N = 40$.
- **Podúloha 4** [**65 bodov**]: $N = 2000$.

V podúlohách 1 a 2 dostane každé riešenie, ktoré správne vyrieši všetky vstupy, plný počet bodov.

V podúlohách 3 a 4 sa počet bodov odvíja od $Q_s$, čo je najväčší počet otázok ktoré vaše riešenie položilo pre niektorý zo zatváracích ceremionálov na vstupoch. Nech $X_s = \max(1, Q_s/N)$. Skóre pre podúlohy 3 a 4 sa potom počíta takto:

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

Hodnota $\text{score}_s$ sa zaokrúhli na najbližšie celé číslo pre každú zo sád a tvoje celkové skóre je ich súčet. Aby si získal plný počet bodov, potrebuješ vyriešiť podúlohu 3 na najviac 55 queries a podúlohu 4

na najviac 2597 queries. Príklady ako sa mení skóre pre sady 3 a 4 v závislosti od hodnoty $Q_s$ sú zobrazené nižšie.

| $Q_s$ | 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 |

## Príklady

| Grader | Riešenie |
|---|---|
| 1<br>5 | |
| | ? 0 2 4 |
| 3 | |
| | ? 3 0 1 |
| 3 | |
| | ? 0 4 3 |
| 5 | |
| | ! 3 1 0 2 4 |

## Vysvetlenie

Vstup v príklade obsahuje jednu testovaciu sadu ($T = 1$) s $N = 5$ účastníčkami. Zasadací poriadok v tejto testovacej sade zodpovedá Obr. 1.

Prvá otázka, ktorú sa ukážkové riešenie spýtalo, je 0, 2, 4. Odpoveď 3 na túto otázku nám vraví, že tieto účastníčky, v nejakom neznámom poradí, sedia na troch po sebe idúcich sedadlách vedľa seba.

Odpoveď 3 na druhú otázku nám hovorí to isté o účastníčkach 3, 0 a 1.

Teraz si už vieme domyslieť, že účastníčka 0 musí sedieť v strede, s účastníčkami 2 a 4 na jednej strane a účastníčkami 1 a 3 na tej druhej.

Po tretej otázke už máme istotu, že účastníčky musia sedieť buď v poradí $[3, 1, 0, 2, 4]$, alebo opačne: $[4, 2, 0, 1, 3]$. Môžeme vypísať hociktorú z týchto dvoch postupností.

## Šablóna programu a detaily evaluácie v CMS

Silne vám odporúčame, aby ste použili poskytnuté šablóny pre C++ a Python. Skontrolujú pre váš či komunikácie s testovačom bola úspešná a ukončia sa ak sa komunikácia preruší.

Ak poskytnuté šablóny nepoužijete, môže sa vám stať, že pre riešenie, ktoré nie je správne, podá CMS nepresný verdikt – napr. namiesto „Output isn't correct“ môžete dostať „Execution killed by signal“ or „Execution timed out (wall clock limit exceeded)“.

Odporúčame vám taktiež použiť testing tool (viď nižšie) na lokálne testovanie vášho riešenie pred submitom. Testing tool skontroluje vaše výstupy a reportuje otázky v zlom formáte.

## Testing tool

Na uľahčenie testovania tvojho riešenia poskytujeme jednoduchý nástroj, ktorý si vieš stiahnuť z CMS. Použitie tohto nástroja je voliteľné. Nezabúdajte, že oficiálny testovať na CMS je niečo iné ako tento testovací nástroj.

Narozdiel od oficiálneho testovača je sample grader *neadaptívny*. To znamená, že mu musíte poskytnúť vstupný súbor. Môžete použiť napríklad ukážkový vstup `seatingplan.input0.txt` alebo si vytvoriť vlastný.

*Vstupný súbor* by mal začínať riadkom s počtom testovacích sád $T$. Každú z $T$ testovacích sád by mali popisovať dva riadky: jeden s číslom $N$ nasledovaný riadom s číslami $g_0, g_1, ..., g_{N-1}$ oddelenými medzerou.

Pre programy v Pythone, povedzme `seatingplan.py` (ktorý bežne spúšťaš ako `pypy3 seatingplan.py`), spusti testovací nástroj takto:

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

Pre C++ programy si svoje riešenie najprv skompiluj:

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

a potom spusti testovací nástroj:

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