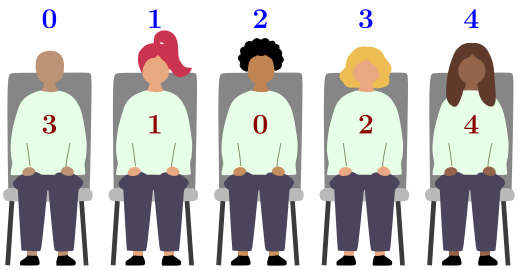
D. Zasedací pořádek (seatingplan)

Závěrečného ceremoniálu letošního EGOI se zúčastní N důležitých hostů. Všichni musí sedět v první řadě v hodně specifickém pořadí, aby se dodržely všechny nuance diplomatického protokolu. Noemi probděla dvě noci, jen aby vymyslela, kdo by měl kde správně sedět.

Veronika na závěrečný ceremoniál dohlíží. Jednou z jejích mnoha povinností je zajistit, aby na sedadlech v první řadě byly správné jmenovky. Má to však háček, Noemi jí správný zasedací pořádek nikdy neřekla a najednou není k nalezení. Naštěstí má fotografka Dorka aplikaci, která by se mohla hodit.

Dorka má pořídit nějaké konkrétní fotky hostů v první řadě, na což si musela připravit foťáky. K tomu potřebovala vědět, jak široká bude každá fotka, a tak jí Noemi napsala aplikaci, která Dorce rychle dá potřebné informace. Veronika teď chce tuto aplikaci použít, aby zjistila správný zasedací pořádek.

Našich N důležitých hostů je očíslováno od 0 do $N-1$. Sedadla v první řadě jsou také očíslovaná od 0 do $N-1$, zleva doprava. Pro každé I ($0 \leq I \leq N-1$) označíme jako g_I hosta, který má sedět na sedadle I, a jako s_I označíme sedadlo, na kterém má sedět host I.



Obrázek 1: Řada s pěti hosty. Pro tuto řadu platí, že $g = [3, 1, 0, 2, 4]$ a $s = [2, 1, 3, 0, 4]$.

Aplikace funguje následovně:

- Dorka zadá čísla I, J, K právě tří různých hostů.
- Aplikace jí řekne minimální počet hostů, kteří budou vidět, pokud budou na fotce všichni tři vybraní hosté. Formálně, aplikace vypíše hodnotu $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Podívejme se třeba na situaci na obrázku Obrázek 1:

- Hosté $I = 0$, $J = 2$ a $K = 4$ sedí na sedadlech $s_I = 2$, $s_J = 3$ a $s_K = 4$. Když si je Dorka vybere, aplikace vypíše hodnotu $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Neboli nejužší fotka, na které jsou hosté 0, 2 a 4, obsahuje jen tyto tři hosty.

- Hosté $I = 0$, $J = 4$ a $K = 3$ sedí na sedadlech $s_I = 2$, $s_J = 4$ a $s_K = 0$. Když si je Dorka vybere, aplikace vypíše hodnotu $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Neboli fotka, na které jsou tito tři hosté, musí obsahovat všech 5 hostů.

Pomozte Veronice zjistit správný zasedací pořádek pomocí Dorčiny aplikace. Přesněji by váš program měl zjistit a vypsat posloupnost $g_0, g_1, ..., g_{N-1}$. Vždy existují právě dvě správné odpovědi (jedna je obrácená verze té druhé), z nichž můžete vypsat kteroukoli. Váš počet bodů bude záviset na tom, kolik dotazů váš program aplikaci položí.

Implementace

⇨ Toto je interaktivní úloha. Váš program bude komunikovat s graderem pomocí standardního vstupu a výstupu ve formátu popsaném níže.

Váš program by měl začít načtením jednoho řádku vstupu, který obsahuje kladné celé číslo T, což je počet problémů, které následují.

Pro každý problém by měl váš program začít načtením jednoho řádku vstupu, který obsahuje kladné celé číslo N, počet sedadel, který je zároveň i počtem hostů.

Pro položení dotazu by měl váš program vypsat řádek ve tvaru „? I J K“, kde $0 \leq I, J, K \leq N-1$ jsou tři **různá** čísla.

Po položení dotazu by měl váš program načíst jeden řádek s jedním kladným celým číslem, což je odpověď na váš dotaz.

Až budete chtít odpovědět, vypište řádek ve tvaru „! g_0 … g_{N-1}“, kde g_0 … g_{N-1} je správný zasedací pořádek.

Po vyřešení všech T problémů by měl váš program normálně skončit.

Berte na vědomí, že oficiální grader použitý v CMS pro testování vašeho řešení může být **adaptivní**. To znamená, že pro některé vstupy není pořadí hostů pevně dané předem. Místo toho se grader může rozhodnout, kterou ze zbývajících permutací používá, podle toho, jaké dotazy vaše řešení už položilo.

Vyprázdnění bufferu (Flushing). Jestli nepoužíváte poskytnuté šablony, nezapomeňte po vypsání každého řádku vyprázdnit standardní výstup (flush), jinak může být váš program ohodnocen jako *Not correct*. V Pythonu se to stane automaticky, pokud používáte `input()` k načítání řádků, a můžete použít `print(..., flush=True)`, abyste vyprázdnění bufferu vynutily. V C++, `cout << endl;` kromě vypsání nového řádku také vyprázdní buffer. Pokud používáte `printf`, použijte `fflush(stdout)`.

Omezení

- $1 \leq T \leq 10$.
- N bude 5 (pouze příklad), 8, 40, nebo 2000.
- Pro každý problém můžete položit maximálně 10 000 dotazů.

Bodování

Váš program bude otestován na několika vstupech rozdělených do podúloh. Pro získání bodů za podúlohu musíte správně vyřešit všechny vstupy, které obsahuje.

- **Podúloha 0 [0 bodů]**: Příklad ($N = 5$).
- **Podúloha 1 [9 bodů]**: $N = 8$.
- **Podúloha 2 [11 bodů]**: $N = 2000$ a hosté 0 a 1 sedí vedle sebe.
- **Podúloha 3 [15 bodů]**: $N = 40$.
- **Podúloha 4 [65 bodů]**: $N = 2000$.

Za podúlohy 1 a 2 dostane plný počet bodů každé řešení, které správně vyřeší všechny testovací vstupy.

V podúlohách 3 a 4 musí vaše řešení správně vyřešit všechny testovací vstupy, abyste získaly vůbec nějaké body. Váš počet bodů pak bude záviset na Q_s, což je největší počet dotazů, které vaše řešení muselo položit k vyřešení jednoho problému v dané podúloze. Nechť $X_s = \max(1, Q_s/N)$. Bodování pro podúlohy 3 a 4 se pak počítají takto:

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

Hodnota skóre_s se v každé podúloze zaokrouhlí na nejbližší celé číslo a váš celkový počet bodů bude jejich součtem. Abyste získaly plný počet bodů, musíte vyřešit podúlohu 3 na nejvýše 55 dotazů a podúlohu 4 na nejvýše 2597 dotazů. Příklady hodnot Q_s a počet bodů za podúlohy 3 a 4 jsou uvedeny níže.

Q_s	55	56	60	70	80	100	150	10000
skóre_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
skóre_4	65	58	53	35	26	21	14	11

Příklady

Grader	Řešení
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Vysvětlení

Ukázkový vstup obsahuje jeden problém ($T = 1$) s $N = 5$ hosty. Skryté pořadí hostů v tomto testovacím vstupu odpovídá obrázku 1.

První dotaz ukázkového řešení je 0, 2, 4. Odpověď 3 na tento dotaz nám říká, že tito hosté sedí v nějakém neznámém pořadí na třech po sobě jdoucích sedadlech vedle sebe.

Odpověď 3 na druhý dotaz nám říká to samé o hostech 3, 0 a 1.

Z toho můžeme odvodit, že host 0 musí sedět uprostřed, přičemž hosté 2 a 4 sedí na jedné straně od hosta 0 a hosté 1 a 3 na druhé.

Po třetím dotazu už si můžeme být jistí, že hosté musí sedět buď v pořadí $[3, 1, 0, 2, 4]$, nebo v obráceném pořadí $[4, 2, 0, 1, 3]$. Můžeme vypsat libovolné z nich.

Šablony kódů a detaily vyhodnocení v CMS

Důrazně doporučujeme používat poskytnuté šablony kódů pro C++ a Python. Ty ověřují, zda byla komunikace s graderem úspěšná, a v případě, že nebyla, korektně program ukončí.

Pokud nepoužijete poskytnuté šablony, může CMS v případech, kdy je vaše řešení nesprávné, zobrazit špatný výsledek. Například namísto „*Output isn't correct*“ (*Nesprávný výstup*) můžete obdržet „*Program byl ukončen signálem*“ (Běh programu ukončen signálem) nebo „*Execution timed out (wall clock limit exceeded)*“ (*Program vyčerpal časový limit (překročený limit na reálný čas)*).

Doporučujeme také využít testovací nástroj (viz níže) k lokálnímu otestování vašeho řešení před jeho odevzdáním. Testovací nástroj kontroluje výstupy vašeho řešení a hlásí případné neplatně dotazy.

Testovací nástroj

Pro usnadnění testování vašeho řešení poskytujeme jednoduchý nástroj, který si můžete stáhnout z CMS. Jeho použití je volitelné. Berte na vědomí, že oficiální grader v CMS je jiný než tento testovací nástroj.

Pro použití tohoto nástroje potřebujete vstupní soubor. Můžete použít poskytnutý ukázkový vstup `seatingplan.input0.txt` nebo si vytvořit svůj vlastní. Vstupní soubor by měl začínat řádkem, na kterém je počet problémů T, a potom by měl obsahovat dva řádky pro každý problém: jeden řádek s číslem N a jeden řádek s čísly $g_0, g_1, ..., g_{N-1}$.

Pro programy v Pythonu, řekněme `seatingplan.py` (které normálně spouštíte jako `pypy3 seatingplan.py`), spusťte testovací nástroj následovně:

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

Pro programy v C++ nejdřív zkompilujte své řešení:

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

a potom spusťte testovací nástroj:

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