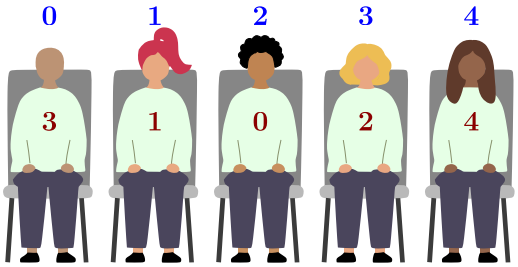
# D. Sēdvietu plāns (seatingplan)

Šī gada EGOI sacensību noslēguma ceremonijā piedalīsies $N$ svarīgi viesi. Viņi jāsasēdina pirmajā rindā ļoti specifiskā secībā, kas atbilst visām diplomātiskā protokola niansēm. Lai noteiktu pareizo sēdvietu secību, Noemi negulēja divas naktis.

Veronika uzrauga noslēguma ceremoniju. Viens no viņas daudzajiem pienākumiem ir pārliecināties, ka uz pirmās rindas krēsliem ir pareizās vārda plāksnītes. Ir tikai viena neliela problēma: Noemi viņai nav pateikusi pareizo sēdvietu secību un tagad nekur nav atrodama. Par laimi fotogrāfei Dorkai ir lietotne, kas varētu būt noderīga.

Dorkai bija jāsagatavo savas fotokameras, lai varētu uzņemt specifiskus fotoattēlus ar viesiem pirmajā rindā. Lai tās pareizi uzstādītu, viņai bija jāzina, cik plats būs katrs fotoattēls, tāpēc Noemi viņai izveidoja lietotni, kas ātri izvada vajadzīgo informāciju. Veronika vēlas izmantot šo lietotni, lai atrastu pareizo sēdvietu izkārtojumu.

$N$ svarīgie viesi ir numurēti no 0 līdz $N-1$. Arī pirmās rindas sēdvietas ir numurētas no 0 līdz $N-1$ no kreisās uz labo pusi. Katram $I$ ($0 \leq I \leq N-1$) ar $g_I$ apzīmēsim viesi, kuram jāsēž sēdvietā $I$, bet ar $s_I$ — sēdvietu, kurā jāsēž viesim $I$.




Attēls 1: Rinda ar pieciem viesiem. Šai rindai $g = [3, 1, 0, 2, 4]$ un $s = [2, 1, 3, 0, 4]$.

Lietotne darbojas šādi:

- Dorka ievada trīs dažādu viesu numurus $I$, $J$, $K$.
- Lietotne pasaka minimālo viesu skaitu, kas būs redzami fotoattēlā, ja tajā jāiekļauj visi trīs izvēlētie viesi. Citiem vārdiem, lietotne parāda vērtību $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Kā piemēru lietosim situāciju, kas parādīta Attēls 1:

- Viesi $I = 0$, $J = 2$ un $K = 4$ sēž sēdvietās $s_I = 2$, $s_J = 3$ un $s_K = 4$. Ja Dorka izvēlēsies šos viesus, lietotne parādīs vērtību $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Citiem vārdiem sakot, šaurākajā fotoattēlā, kurā ir iekļauti viesi 0, 2 un 4, būs redzami tikai viņi trīs.

- Viesi $I = 0$, $J = 4$ un $K = 3$ sēž sēdvietās $s_I = 2$, $s_J = 4$ un $s_K = 0$. Ja Dorka izvēlēsies šos viesus, lietotne parādīs vērtību $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Citiem vārdiem sakot, fotoattēlā, kurā jāiekļauj šie trīs viesi, jābūt redzamiem visiem 5 viesiem.

Palīdzi Veronikai noteikt pareizo sēdvietu secību, izmantojot Dorkas lietotni. Precīzāk sakot, tavai programmai jānosaka un jāizvada secība $g_0, g_1, ..., g_{N-1}$. Vienmēr ir tieši divas pareizas atbildes (viena ir otras apgrieztā secība), un tu vari izvadīt jebkuru no tām. Tavs rezultāts būs atkarīgs no tā, cik daudz vaicājumu lietotnei tava programma veiks.

## Implementācija

⇨ Šis ir interaktīvs uzdevums. Tava programma izmantos standarta ievadi un izvadi, lai sazinātos ar vērtētāju tālāk aprakstītajā formātā.

Tavai programmai jāsāk ar pirmās ievaddatu rindas nolasīšanu, kurā dots pozitīvs vesels skaitlis $T$ — testu skaits.

Katrā testā programmai vispirms jānolasa viena rinda ar pozitīvu veselu skaitli $N$ — sēdvietu skaitu, kas vienlaikus ir arī viesu skaits.

Lai veiktu vaicājumu, tavai programmai jāizvada rinda formātā „? $I$ $J$ $K$“, kur $0 \leq I, J, K \leq N-1$ ir trīs **dažādi** skaitļi.

Pēc vaicājuma veikšanas tavai programmai jānolasa viena rinda, kurā dots viens pozitīvs vesels skaitlis — atbilde uz tavu vaicājumu.

Lai sniegtu atbildi ar pareizo sēdvietu secību, tavai programmai jāizvada rinda formātā „! $g_0$ … $g_{N-1}$“.

Pēc visu $T$ testu izpildes programmai jābeidz darbs.

Ņem vērā, ka oficiālais CMS vērtētājs, kas tiek izmantots tava risinājuma pārbaudei, var būt **adaptīvs**. Tas nozīmē, ka dažos testos viesu secība nav noteikta iepriekš. Tā vietā vērtētājs var izvēlēties, kuru no vēl iespējamajām permutācijām izmantot, atkarībā no vaicājumiem, ko tava programma jau ir veikusi.

**Izvada bufera iztukšošana.** Ja neizmanto nodrošinātās koda sagatavnes, pārliecinies, ka pēc katras rindas izvadīšanas tiek iztukšots standarta izvades buferis. Pretējā gadījumā tavs risinājums var tikt novērtēts kā *Not correct*. Python valodā tas notiek automātiski, ja rindu nolasīšanai izmanto `input()`, kā arī var izmantot `print(..., flush=True)`. C++ valodā `cout << endl;` iztukšo buferi un arī izvada jaunu rindiņu; ja izmanto `printf`, lieto `fflush(stdout)`.

## Ierobežojumi

- $1 \leq T \leq 10$.
- $N$ būs 5 (tikai piemērā), 8, 40 vai 2000.
- Katrā testā drīkst veikt ne vairāk kā 10 000 vaicājumus.

## Vērtēšana

Tava programma tiks pārbaudīta ar vairākiem testiem, kas sagrupēti apakšuzdevumos. Lai iegūtu punktus apakšuzdevumā, tev pareizi jāatrisina visi tajā iekļautie testi.

- **Apakšuzdevums 0 [ 0 punkti]**: Piemērs ($N = 5$).
- **Apakšuzdevums 1 [ 9 punkti]**: $N = 8$.
- **Apakšuzdevums 2 [11 punkti]**: $N = 2000$ un viesi 0 un 1 sēž blakus.
- **Apakšuzdevums 3 [15 punkti]**: $N = 40$.
- **Apakšuzdevums 4 [65 punkti]**: $N = 2000$.

1. un 2. apakšuzdevumā jebkurš risinājums, kas pareizi atrisina visus testus, saņems visus punktus.

3. un 4. apakšuzdevumā tavam risinājumam pareizi jāatrisina visi testi, un tavs vērtējums būs atkarīgs no $Q_s$ — lielākā vaicājumu skaita, kas tavam risinājumam bija nepieciešams kāda testa atrisināšanai.

Apzīmēsim $X_s = \max(1, Q_s/N)$. Punkti par 3. un 4. apakšuzdevumu tiek aprēķināti šādi:

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

Vērtība $\text{score}_s$ katram apakšuzdevumam tiek noapaļota līdz tuvākajam veselam skaitlim, un tavs kopējais rezultāts ir šo punktu summa. Lai iegūtu maksimālo punktu skaitu, 3. apakšuzdevums jāatrisina ar ne vairāk kā 55 vaicājumiem, bet 4. apakšuzdevums — ar ne vairāk kā 2597 vaicājumiem. Zemāk redzamas $Q_s$ vērtības un atbilstošie punkti 3. un 4. apakšuzdevumam.

| $Q_s$ | 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 |

## Piemēri

| Graders | Atrisinājums |
|---|---|
| 1<br>5 | |
| | ? 0 2 4 |
| 3 | |
| | ? 3 0 1 |
| 3 | |
| | ? 0 4 3 |
| 5 | |
| | ! 3 1 0 2 4 |

## Skaidrojums

Piemērā ir viens tests ($T = 1$) ar $N = 5$ viesiem. Šajā testā slēptā viesu konfigurācija atbilst Attēls 1.

Piemēra risinājuma pirmais vaicājums ir 0, 2, 4. Tā kā atbilde uz šo vaicājumu ir 3, varam secināt, ka šie viesi sēž trijās blakusesošās sēdvietās nezināmā secībā.

Otrajā vaicājumā atbilde 3 pasaka to pašu par viesiem 3, 0 un 1.

Tagad varam secināt, ka viesim 0 jāsēž vidū, kamēr viesi 2 un 4 ir vienā pusē, bet viesi 1 un 3 — otrā pusē.

Pēc trešā vaicājuma jau varam būt droši, ka viesiem jāsēž vai nu secībā $[3, 1, 0, 2, 4]$, vai arī apgrieztajā secībā $[4, 2, 0, 1, 3]$. Derīga ir jebkura no šīm secībām.

## Koda sagatavnes un CMS vērtēšana

Silti iesakām izmantot piedāvātās koda sagatavnes C++ un Python valodām. Tās pārbauda, vai saziņa ar vērtētāju ir bijusi veiksmīga un korekti beidz darbu, ja tā nav bijusi veiksmīga.

Ja neizmantosi piedāvātās sagatavnes, gadījumos, kad tavs risinājums būs nepareizs, CMS var parādīt nepareizu vērtējumu. Piemēram, ziņojuma "Output isn't correct" vietā vari saņemt ziņojumu "Execution killed by signal" vai "Execution timed out (wall clock limit exceeded)".

Iesakām arī pirms risinājuma iesūtīšanas to lokāli pārbaudīt, izmantojot testēšanas rīku (skatīt zemāk). Testēšanas rīks pārbauda tava risinājuma izvaddatus un ziņo par protokola pārkāpumiem.

## Testēšanas rīks

Lai atvieglotu tava risinājuma testēšanu, tiek piedāvāts vienkāršs rīks, kuru var lejupielādēt no CMS. Rīka izmantošana nav obligāta. Ņem vērā, ka oficiālais CMS vērtētājs atšķiras no šī testēšanas rīka.

Lai izmantotu rīku, tev nepieciešams ievades fails. Vari izmantot piedāvāto ievaddatu failu `seatingplan.input0.txt` vai izveidot savu. Ievades failam jāsākas ar rindu, kurā dots testu skaits $T$, un pēc tam katrā testā jābūt divām rindām: pirmajā rindā jābūt skaitlim $N$, un otrajā rindā jābūt skaitļiem $g_0, g_1, ..., g_{N-1}$.

Python programmām, piemēram, `seatingplan.py` (parasti palaiž ar `pypy3 seatingplan.py`), palaid testēšanas rīku šādi:

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

C++ programmām vispirms nokompilē savu risinājumu:

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

un tad palaid testēšanas rīku:

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