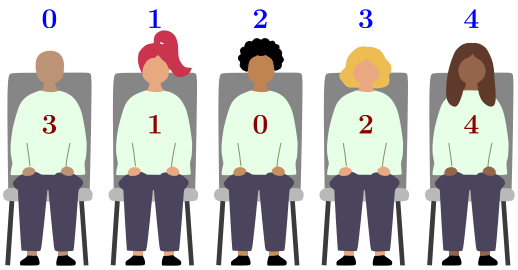
# D. Raspored sjedenja (seatingplan)

Svečanosti zatvaranja ovogodišnjeg EGOI-ja prisustvovat će $N$ važnih uzvanika. Svi oni moraju sjediti u prvom redu točno određenim redoslijedom koji odgovara svim nijansama diplomatskog protokola. Određivanje ispravnog rasporeda sjedenja Mr. Malnaru oduzelo je dvije neprospavane noći.

Sydney nadgleda svečanost zatvaranja. Jedna od njezinih brojnih odgovornosti je osigurati da sjedala u prvom redu imaju ispravne natpise s imenima. Postoji samo jedan mali problem: Mr. Malnar joj nikada nije rekao točan raspored sjedenja, a sada Mr. Malnaru nema ni traga. Srećom, fotografkinja Sweeney ima aplikaciju koja bi mogla biti korisna.

Sweeney je morala pripremiti svoje fotoaparate kako bi mogla snimiti neke specifične fotografije uzvanika u prvom redu. Za postavljanje je trebala znati kolika će biti širina svake fotografije, pa joj je Mr. Malnar izradio aplikaciju koja brzo izbacuje informacije koje su joj potrebne. Sydney sada želi koristiti tu aplikaciju kako bi pronašla ispravan raspored sjedenja.

$N$ važnih uzvanika označeni su brojevima od 0 do $N-1$. Sjedala u prvom redu također su označena brojevima od 0 do $N-1$, s lijeva na desno. Za svaki $I$ ($0 \leq I \leq N-1$), neka $g_I$ označava uzvanika koji bi trebao sjediti na sjedalu $I$, a neka $s_I$ označava sjedalo na kojem bi gost $I$ trebao sjediti.




Slika 1: Red s pet uzvanika. Za ovaj red, $g = [3, 1, 0, 2, 4]$ i $s = [2, 1, 3, 0, 4]$.

Aplikacija radi na sljedeći način:

- Sweeney unosi brojeve $I$, $J$, $K$ točno tri različita uzvanika.
- Aplikacija joj javlja minimalan broj uzvanika koji će biti vidljiv ako su sva tri odabrana uzvanika na fotografiji. Formalno, aplikacija će prikazati vrijednost $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Na primjer, pogledajte situaciju prikazanu na Slika 1:

- Uzvanici $I = 0$, $J = 2$ i $K = 4$ nalaze se na sjedalima $s_I = 2$, $s_J = 3$ i $s_K = 4$. Ako ih Sweeney odabere, aplikacija će prikazati vrijednost $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Drugim riječima, najuža fotografija koja sadrži uzvanike 0, 2 i 4 ima samo ta tri uzvanika.

- Uzvanici $I = 0$, $J = 4$ i $K = 3$ nalaze se na sjedalima $s_I = 2$, $s_J = 4$ i $s_K = 0$. Ako ih Sweeney odabere, aplikacija će prikazati vrijednost $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Drugim riječima, fotografija koja sadrži ta tri zadana uzvanika mora sadržavati svih 5 uzvanika.

Pomozite Sydney odrediti ispravan raspored sjedenja koristeći Sweeneyinu aplikaciju. Točnije, vaš program treba odrediti i ispisati niz $g_0, g_1, ..., g_{N-1}$. Uvijek postoje točno dva ispravna odgovora (jedan je obrnuti poredak drugog), a vi možete ispisati bilo kojeg od njih. Vaš rezultat ovisit će o broju upita koje vaša otopina uputi aplikaciji.

## Implementacija

⇨ Ovo je interaktivni problem. Vaš program koristit će standardni ulaz i izlaz za komunikaciju s ocjenjivačem u formatu opisanom u nastavku.

Vaš program treba započeti čitanjem jednog retka ulaza koji sadrži pozitivan cijeli broj $T$, broj test primjera koji slijede.

Za svaki test primjer, vaš program treba započeti čitanjem jednog retka ulaza koji sadrži pozitivan cijeli broj $N$, broj sjedala, što je ujedno i broj uzvanika.

Kako biste napravili upit, vaš program treba ispisati redak oblika „? $I$ $J$ $K$", gdje su $0 \leq I, J, K \leq N-1$ tri **različita** broja.

Nakon upita, vaš program treba pročitati jedan redak koji sadrži jedan pozitivan cijeli broj, odgovor na vaš upit.

Kako biste odgovorili s točnim rasporedom sjedenja, vaš program treba ispisati redak oblika „! $g_0$ … $g_{N-1}$".

Nakon rješavanja svih $T$ test primjera, vaš program treba normalno završiti s radom.

Napomena: službeni ocjenjivač koji se koristi u CMS-u za testiranje vašeg rješenja može biti **prilagodljiv** (adaptive). To jest, za neke test primjere, permutacija uzvanika nije određena unaprijed. Umjesto toga, ocjenjivač može odlučiti koju od preostalih permutacija koristiti ovisno o upitima koje je vaš program već postavio.

**Ispiranje izlaza (Flushing).** Ako ne koristite ponuđene predloške, pobrinite se da isperete standardni izlaz nakon ispisa svakog retka, inače bi vaš program mogao biti ocijenjen kao *Netočan*. U Pythonu se to događa automatski ako koristite `input()` za čitanje redaka, a možete koristiti `print(..., flush=True)` kako biste prisilili ispiranje. U C++-u, `cout << endl;` ispire izlaz uz ispis znaka novog retka; ako koristite printf, upotrijebite `fflush(stdout)`.

## Ograničenja

- $1 \leq T \leq 10$.
- $N$ će biti 5 (samo primjer), 8, 40 ili 2000.
- Za svaki test primjer, možete postaviti najviše 10 000 upita.

## Bodovanje

Vaš program bit će testiran na nekoliko test primjera grupiranih u podzadatke. Kako biste dobili bodove za podzadatak, morate točno riješiti sve testove koje on sadrži.

- **Podzadatak 0 [ 0 bodova]**: Primjer ($N = 5$).
- **Podzadatak 1 [ 9 bodova]**: $N = 8$.
- **Podzadatak 2 [11 bodova]**: $N = 2000$, a uzvanici 0 i 1 sjede jedan pored drugog.
- **Podzadatak 3 [15 bodova]**: $N = 40$.
- **Podzadatak 4 [65 bodova]**: $N = 2000$.

Za podzadatke 1 i 2, svako rješenje koje točno riješi sve test primjere dobit će sve bodove.

Za podzadatke 3 i 4, vaše rješenje mora točno riješiti sve test primjere kako bi osvojilo bilo kakve bodove, a vaš rezultat ovisit će o $Q_s$, najvećem broju upita koje je vaše rješenje moralo postaviti kako bi riješilo test primjer. Neka je $X_s = \max(1, Q_s/N)$. Bodovi za podzadatke 3 i 4 tada se izračunavaju kako slijedi:

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

Vrijednost $\text{score}_s$ zaokružuje se na najbliži cijeli broj po podzadatku, a vaš ukupni rezultat je zbroj tih vrijednosti. Kako biste dobili maksimalan broj bodova, trebate riješiti podzadatak 3 s najviše 55 upita, a podzadatak 4 s najviše 2597 upita. Primjeri vrijednosti $Q_s$ i rezultata za podzadatke 3 i 4 prikazani su u nastavku.

| $Q_s$ | 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 |

## Primjeri

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

## Objašnjenje

Ulaz iz primjera sadrži jedan test primjer ($T = 1$) s $N = 5$ uzvanika. Skriveni raspored uzvanika u ovom test primjeru odgovara onom na Slika 1.

Prvi upit koji je postavilo ogledno rješenje je 0, 2, 4. Odgovor 3 na ovaj upit govori nam da ti uzvanici sjede, nepoznatim redoslijedom, na tri susjedna sjedala jedno pored drugog.

Odgovor 3 na drugi upit govori nam isto o uzvanicima 3, 0 i 1.

Sada možemo zaključiti da uzvanik 0 mora sjediti u sredini, s uzvanicima 2 i 4 na jednoj strani, te uzvanicima 1 i 3 na drugoj strani.

Nakon trećeg upita već možemo biti sigurni da uzvanici moraju sjediti ili u poretku $[3, 1, 0, 2, 4]$ ili u obrnutom poretku $[4, 2, 0, 1, 3]$. Možemo ispisati bilo koji od ta dva poretka.

## Kôdni predlošci i detalji ocjenjivanja u CMS-u

Snažno preporučujemo korištenje priloženih kôdnih predložaka za C++ i Python. Oni provjeravaju je li komunikacija s ocjenjivačem bila uspješna i graciozno završavaju rad ako nije.

Ako ne koristite ponuđene predloške, u slučajevima kada vaše rješenje nije točno, CMS bi mogao prikazati pogrešnu presudu. Na primjer, umjesto „Output isn't correct" mogli biste dobiti „Execution killed by signal" ili „Execution timed out (wall clock limit exceeded)".

Također preporučujemo alat za testiranje (vidi dolje) kako biste lokalno testirali svoje rješenje prije predaje. Alat za testiranje provjerava izlaze vašeg rješenja i prijavljuje korištenje nevaljanih upita.

## Alat za testiranje

Kako bismo olakšali testiranje vašeg rješenja, nudimo jednostavan alat koji možete preuzeti s CMS--a. Korištenje alata je opcionalno. Napomena: službeni ocjenjivač na CMS-u razlikuje se od alata za testiranje.

Za korištenje alata trebate ulaznu datoteku. Možete koristiti priloženi primjer ulaza `seatingplan.input0.txt` ili napraviti svoj. Ulazna datoteka treba započeti retkom koji sadrži broj $T$ test primjera, a zatim bi trebala imati dva retka po test primjeru: jedan redak s brojem $N$, a zatim jedan redak s brojevima $g_0, g_1, ..., g_{N-1}$.

Za Python programe, recimo `seatingplan.py` (obično se pokreće kao `pypy3 seatingplan.py`) pokrenite alat za testiranje na sljedeći način:

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

Za C++ programe, prvo prevedite svoje rješenje:

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

a zatim pokrenite alat za testiranje:

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