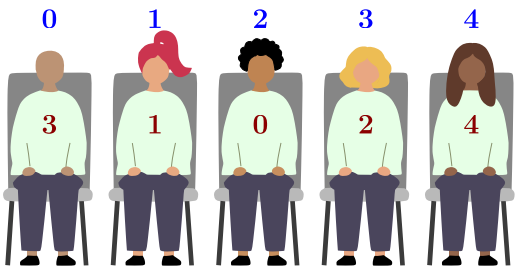
D. Plan sjedenja (seatingplan)

Ceremoniji zatvaranja ovogodišnjeg EGOI-ja prisustvovat će N važnih gostiju. Svi oni moraju biti smješteni u prvi red vrlo specifičnim redoslijedom koji odgovara svim nijansama diplomatskog protokola. Određivanje ispravnog redoslijeda sjedenja Noemiju je koštalo dvije neprospavane noći.

Veronica nadgleda ceremoniju zatvaranja. Jedna od njenih mnogih odgovornosti je da osigura da sjedišta u prvom redu imaju ispravne natpise s imenima. Postoji samo jedan mali problem: Noemi joj nikada nije rekla ispravan redoslijed sjedenja, a sada ju ne može nigdje pronaći. Srećom, fotografkinja Dorka ima aplikaciju koja bi mogla biti korisna.

Dorka je morala pripremiti svoje kamere kako bi mogla napraviti neke specifične fotografije gostiju u prvom redu. Za postavljanje, morala je znati koliko će široka biti svaka fotografija, pa joj je Noemi napravila aplikaciju koja brzo daje informacije koje su joj potrebne. Veronica sada želi koristiti aplikaciju kako bi pronašla ispravan raspored sjedenja.

N važnih gostiju označeno je brojevima od 0 do $N-1$. Sjedišta u prvom redu su također označena brojevima od 0 do $N-1$, slijeva nadesno. Za svaki I $(0 \leq I \leq N-1)$, neka g_I označava gosta koji treba sjediti na sjedištu I, a neka s_I označava sjedište na kojem gost I treba sjediti.



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

Aplikacija radi na sljedeći način:

- Dorka unosi brojeve I, J, K za tri različita gosta.
- Aplikacija joj govori minimalan broj gostiju koji će biti vidljivi ako su sva tri odabrana gosta 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:

- Gosti $I = 0$, $J = 2$ i $K = 4$ su na sjedištima $s_I = 2$, $s_J = 3$ i $s_K = 4$. Ako ih Dorka odabere, aplikacija će prikazati vrijednost $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Drugim riječima, najuža fotografija koja sadrži goste 0, 2 i 4 ima samo ova tri gosta.

- Gosti $I = 0$, $J = 4$ i $K = 3$ su na sjedištima $s_I = 2$, $s_J = 4$ i $s_K = 0$. Ako ih Dorka odabere, aplikacija će prikazati vrijednost $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Drugim riječima, fotografija koja sadrži trojicu navedenih gostiju mora sadržavati svih 5 gostiju.

Pomozite Veronici da odredi ispravan redoslijed sjedenja koristeći Dorkinu aplikaciju. Preciznije, vaš program treba odrediti i ispisati niz $g_0, g_1, ..., g_{N-1}$. Uvijek postoje tačno dva tačna odgovora (jedan je obrnuti od drugog), a vi možete ispisati bilo koji od njih. Vaš rezultat će zavisiti od broja upita koje vaše rješenje postavi aplikaciji.

Implementacija

⇨ Ovo je interaktivni zadatak. Vaš program će koristiti standardni ulaz i izlaz za komunikaciju sa graderom u formatu opisanom u nastavku.

Vaš program treba započeti čitanjem jedne linije ulaza koja sadrži pozitivan cijeli broj T, broj test primjera koji slijede.

Za svaki test primjer, vaš program treba započeti čitanjem jedne linije ulaza koja sadrži pozitivan cijeli broj N, broj sjedišta, što je ujedno i broj gostiju.

Da biste postavili upit, vaš program treba ispisati liniju formata "? I J K", gdje su $0 \leq I, J, K \leq N-1$ tri **različita** broja.

Nakon postavljanja upita, vaš program treba pročitati jednu liniju koja sadrži jedan pozitivan cijeli broj, odgovor na vaš upit.

Da biste odgovorili tačnim redoslijedom sjedenja, vaš program treba ispisati liniju formata "! g_0 … g_{N-1}".

Nakon rješavanja svih T test primjera, vaš program bi se trebao normalno završiti.

Imajte na umu da službeni grader koji se koristi u CMS-u za testiranje vašeg rješenja može biti **adaptivan**. To jest, za neke test primjere, permutacija gostiju nije unaprijed određena. Umjesto toga, grader može odlučiti koju od preostalih permutacija koristi ovisno o upitima koje je vaš program već postavio.

Pražnjenje (Flushing). Ako ne koristite ponuđene predloške, provjerite jeste li ispraznili standardni izlaz (flush) nakon ispisa svakog reda, inače bi vaše rješenje moglo biti ocijenjeno kao *Not correct* (Netačno). U Pythonu se ovo događa automatski ako koristite `input()` za čitanje redova, a možete koristiti `print(..., flush=True)` da prisilite pražnjenje. U C++, `cout << endl;` prazni izlaz pored ispisa novog reda; ako koristite printf, koristite `fflush(stdout)`.

Ograničenja

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

Bodovanje

Vaš program će se testirati na nekoliko test primjera grupisanih u podzadatke. Da biste dobili bodove za podzadatak, morate tačno riješiti sve testove koje on sadrži.

- **Podzadatak 0** [**0 bodova**]: Primjer ($N = 5$).
- **Podzadatak 1** [**9 bodova**]: $N = 8$.
- **Podzadatak 2** [**11 bodova**]: $N = 2000$, i gosti 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 tačno riješi sve test primjere dobit će sve bodove.

Za podzadatke 3 i 4 vaše rješenje mora tačno riješiti sve test primjere da bi osvojilo bilo kakve bodove, a vaš rezultat će zavisiti od Q_s, najvećeg broja upita koje je vaše rješenje trebalo napraviti da riješi test primjer. Neka je $X_s = \max(1, Q_s/N)$. Rezultati za podzadatke 3 i 4 se tada računaju 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 score_s se zaokružuje na najbliži cijeli broj po podzadatku, a vaš ukupan rezultat je zbir ovih vrijednosti. Da biste dobili maksimalan broj bodova, morate riješiti podzadatak 3 sa najviše 55 upita, a podzadatak 4 sa najviše 2597 upita. Primjeri vrijednosti Q_s i bodova za podzadatke 3 i 4 su prikazani ispod.

Q_s	55	56	60	70	80	100	150	10000
score_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
score_4	65	58	53	35	26	21	14	11

Primjeri ulaza/izlaza

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

Objašnjenje

Primjer ulaza sadrži jedan test primjer ($T = 1$) sa $N = 5$ gostiju. Skrivena konfiguracija gostiju u ovom test primjeru odgovara Slika 1.

Prvi upit koji je napravilo rješenje iz primjera je 0, 2, 4. Odgovor 3 na ovaj upit nam govori da ti gosti sjede, nekim nepoznatim redoslijedom, na tri uzastopna sjedišta jedno pored drugog.

Odgovor 3 na drugi upit nam govori istu stvar o gostima 3, 0 i 1.

Sada možemo zaključiti da gost 0 mora sjediti u sredini, sa gostima 2 i 4 na jednoj strani, a gostima 1 i 3 na drugoj strani.

Nakon trećeg upita, već možemo biti sigurni da gosti moraju sjediti ili u redoslijedu $[3, 1, 0, 2, 4]$ ili u obrnutom redoslijedu $[4, 2, 0, 1, 3]$. Možemo ispisati bilo koji od ova dva redoslijeda.

Predlošci koda i detalji ocjenjivanja u CMS-u

Toplo preporučujemo korištenje priloženih predložaka koda za C++ i Python. Oni provjeravaju je li komunikacija sa graderom bila uspješna i završavaju rad na pristojan način kada nije.

Ako ne koristite priložene predloške, u slučajevima kada je vaše rješenje netačno, CMS može prikazati pogrešnu presudu. Na primjer, umjesto "Output isn't correct" možete dobiti "Execution killed by signal" ili "Execution timed out (wall clock limit exceeded)".

Također preporučujemo alat za testiranje (vidi dolje) kako biste testirali svoje rješenje lokalno prije slanja. Alat za testiranje provjerava izlaze vašeg rješenja i javlja upotrebu nevažećih upita.

Alat za testiranje

Kako bismo olakšali testiranje vašeg rješenja, nudimo jednostavan alat koji možete preuzeti sa CMS-a. Korištenje alata je opcionalno. Imajte na umu da je službeni grader na CMS-u drugačiji od alata za testiranje.

Da biste koristili alat, potreban vam je ulazni fajl. Možete koristiti priloženi primjer ulaza `seatingplan.input0.txt` ili napraviti svoj. Ulazni fajl treba početi linijom koja ima broj T test primjera, a zatim treba imati dvije linije po test primjeru: jednu liniju sa brojem N i jednu liniju sa 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 kompajlirajte 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
```