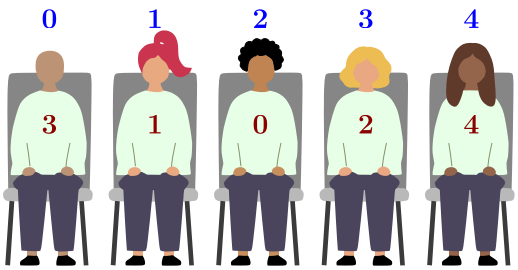
D. Razporeditev sedežev (seatingplan)

Zaključne slovesnosti EGOI se bo udeležilo N pomembnih gostov. Vsi morajo sedeti v prvi vrsti v zelo določenem vrstnem redu, ki ustreza vsem niansam diplomatskega protokola. Noemi je za določitev pravilnega vrstnega reda potrebovala dve neprespani noči.

Veronica nadzoruje zaključno slovesnost. Ena od njenih številnih nalog je, da poskrbi, da imajo sedeži v prvi vrsti pravilne napise z imeni. Obstaja le ena majhna težava: Noemi ji nikoli ni povedala pravilnega vrstnega reda, zdaj pa Noemi nikjer ni mogoče najti. Na srečo ima fotografinja Dorka aplikacijo, ki bi lahko bila koristna.

Dorka je morala pripraviti svoje kamere, da bi lahko posnela nekaj posebnih fotografij gostov v prvi vrsti. Za nastavitev je morala vedeti, kako široka bo vsaka fotografija, zato ji je Noemi izdelala aplikacijo, ki hitro izpiše podatke, ki jih potrebuje. Veronica želi zdaj uporabiti aplikacijo, da bi ugotovila pravilno razporeditev sedežev.

Pomembni gostje so oštevilčeni od 0 do $N-1$. Sedeži v prvi vrsti so prav tako oštevilčeni od 0 do $N-1$, od leve proti desni. Za vsak I ($0 \leq I \leq N-1$) naj g_I označuje gosta, ki mora sedeti na sedežu I, s_I pa naj označuje sedež, na katerem mora sedeti gost I.



Slika 1: Vrsta s petimi gosti. Za to vrsto je $g = [3, 1, 0, 2, 4]$ in $s = [2, 1, 3, 0, 4]$.

Aplikacija deluje takole:

- Dorka vnese številke I, J, K točno treh različnih gostov.
- Aplikacija ji pove najmanjše število gostov, ki bodo vidni, če so vsi trije izbrani gostje na fotografiji. Formalno bo aplikacija prikazala vrednost $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Za primer glej situacijo, prikazano na Slika 1:

- Gostje $I = 0$, $J = 2$ in $K = 4$ so na sedežih $s_I = 2$, $s_J = 3$ in $s_K = 4$. Če jih Dorka izbere, bo aplikacija prikazala vrednost $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Z drugimi besedami, najožja fotografija, ki vsebuje goste 0, 2 in 4, vsebuje samo te tri goste.

- Gostje $I = 0$, $J = 4$ in $K = 3$ so na sedežih $s_I = 2$, $s_J = 4$ in $s_K = 0$. Če jih Dorka izbere, bo aplikacija prikazala vrednost $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Z drugimi besedami, fotografija, ki vsebuje tri podane goste, mora vsebovati vseh 5 gostov.

Pomagaj Veronici določiti pravilni vrstni red sedežev z uporabo Dorkine aplikacije. Natančneje, tvoj program mora določiti in izpisati zaporedje $g_0, g_1, ..., g_{N-1}$. Vedno obstajata točno dva pravilna odgovora (eden je obraten od drugega), izpišeš lahko katerega koli od njiju. Tvoj rezultat bo odvisen od števila poizvedb, ki jih tvoja rešitev opravi v aplikaciji.

Implementacija

⇨ To je interaktivna naloga. Tvoj program bo za komunikacijo z ocenjevalnikom uporabljal standardni vhod in izhod v spodaj opisanem formatu.

Tvoj program naj najprej prebere eno vrstico vhoda, ki vsebuje pozitivno celo število T, število testnih primerov, ki sledijo.

Za vsak testni primer mora tvoj program začeti z branjem ene vrstice vhoda, ki vsebuje pozitivno celo število N, število sedežev, kar je tudi število gostov.

Za poizvedbo naj tvoj program izpiše vrstico oblike „? I J K“, kjer so $0 \leq I, J, K \leq N-1$ tri **različne** številke.

Po izvedbi poizvedbe mora tvoj program prebrati eno vrstico, ki vsebuje eno pozitivno celo število, odgovor na tvojo poizvedbo.

Za odgovor s pravilno razporeditvijo sedežev mora tvoj program izpisati vrstico oblike „! g_0 … g_{N-1}“.

Po rešitvi vseh T testnih primerov se mora tvoj program normalno zaključiti.

Upoštevaj, da je lahko uradni ocenjevalnik v CMS, ki preverja tvojo rešitev, **prilagodljiv** (ang. adaptive). To pomeni, da za nekatere testne primere permutacija gostov ni določena vnaprej. Ocenjevalnik se lahko namesto tega odloči, katero od preostalih permutacij bo uporabil, odvisno od poizvedb, ki jih je tvoj program že postavil.

Izpiranje (flushing). Če ne uporabljaš priloženih predlog, poskrbi, da po izpisu vsake vrstice izpraznišš (flush) standardni izhod, sicer bo tvoj program morda ocenjen kot *Nepravilen*. V jeziku Python se to zgodi samodejno, če za branje vrstic uporabljaš `input()`, za prisilno izpiranje pa lahko uporabiš `print(..., flush=True)`. V jeziku C++ `cout << endl;` poleg izpisa nove vrstice tudi izvede izpiranje; pri uporabi printf uporabi `fflush(stdout)`.

Omejitve

- $1 \leq T \leq 10$.
- N bo 5 (samo primer), 8, 40 ali 2000.
- Za vsak testni primer lahko izvedeš največ 10 000 poizvedb.

Točkovanje

Tvoja rešitev bo testirana na več testnih primerih, razdeljenih v podnaloge. Da prejmeš točke za podnalogo, mora tvoj program pravilno rešiti vse teste, ki jih vsebuje podnaloga.

- **Podnaloga 0 [0 točk]**: Primer ($N = 5$).
- **Podnaloga 1 [9 točk]**: $N = 8$.
- **Podnaloga 2 [11 točk]**: $N = 2000$, in gosta 0 in 1 sedita drug ob drugem.
- **Podnaloga 3 [15 točk]**: $N = 40$.
- **Podnaloga 4 [65 točk]**: $N = 2000$.

Za podnalogi 1 in 2 bo vsaka rešitev, ki pravilno reši vse testne primere, prejela vse točke.

Za podnalogi 3 in 4 mora tvoja rešitev pravilno rešiti vse testne primere, da sploh dobiš točke, tvoj rezultat pa bo odvisen od Q_s, največjega števila poizvedb, ki jih je tvoja rešitev potrebovala za rešitev testnega primera. Naj bo $X_s = \max(1, Q_s/N)$. Rezultati za podnalogi 3 in 4 se nato izračunajo takole:

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

Vrednost score_s se zaokroži na najbližje celo število na podnalogo, tvoj skupni rezultat pa je vsota teh. Da bi dobila vse točke, moraš podnalogo 3 rešiti z največ 55 poizvedbami, podnalogo 4 pa z največ 2597 poizvedbami. Primeri vrednosti Q_s in rezultatov za podnalogi 3 in 4 so prikazani spodaj.

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

Primeri vhoda/izhoda

Ocenjevalnik	Rešitev
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Razlaga

Vhodni primer vsebuje en testni primer ($T = 1$) z $N = 5$ gosti. Skrita konfiguracija gostov v tem testnem primeru ustreza Slika 1.

Prva poizvedba, ki jo je opravila zgledna rešitev, je 0, 2, 4. Odgovor 3 na to poizvedbo nam pove, da ti gostje sedijo v nekem neznanem vrstnem redu na treh zaporednih sedežih drug ob drugem.

Odgovor 3 na drugo poizvedbo nam pove isto o gostih 3, 0 in 1.

Zdaj lahko sklepamo, da mora gost 0 sedeti na sredini, z gostoma 2 in 4 na eni strani ter gostoma 1 in 3 na drugi strani.

Po tretji poizvedbi smo lahko že prepričani, da morajo gostje sedeti bodisi v vrstnem redu $[3, 1, 0, 2, 4]$ bodisi v obrnjenem vrstnem redu $[4, 2, 0, 1, 3]$. Izpišemo lahko katero koli od teh zaporedij.

Predloge kode in podrobnosti ocenjevanja v CMS

Močno priporočamo uporabo priloženih predlog kode za C++ in Python. Te preverijo, ali je bila komunikacija z ocenjevalnikom uspešna, in se elegantno zaključijo, če ni bila.

Če ne uporabljaš priloženih predlog, CMS v primerih, ko tvoja rešitev ni pravilna, morda prikaže napačno razsodbo. Na primer, namesto „Output isn't correct“ (Izhod ni pravilen) lahko prejmeš „Execution killed by signal“ ali „Execution timed out (wall clock limit exceeded)“.

Priporočamo tudi orodje za testiranje (glej spodaj), s katerim svojo rešitev testiraš lokalno, preden jo oddaš. Orodje za testiranje preveri izhode tvoje rešitve in poroča o uporabi neveljavnih poizvedb.

Orodje za testiranje

Da bi olajšali testiranje tvoje rešitve, nudimo preprosto orodje, ki ga lahko preneseš iz CMS. Uporaba orodja ni obvezna. Upoštevaj, da je uradni ocenjevalnik v CMS drugačen od orodja za testiranje.

Za uporabo orodja potrebuješ vhodno datoteko. Uporabiš lahko priložen vhodni primer `seatingplan.input0.txt` ali pa narediš svojega. Vhodna datoteka se mora začeti z vrstico, ki vsebuje število T testnih primerov, nato pa mora imeti dve vrstici na testni primer: eno vrstico s številom N in nato eno vrstico s številkami $g_0, g_1, ..., g_{N-1}$.

Za programe v jeziku Python, recimo `seatingplan.py` (običajno se zažene kot `pypy3 seatingplan.py`), orodje za testiranje zaženeš takole:

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

Za programe v jeziku C++ najprej prevedeš svojo rešitev:

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

in nato zaženeš orodje za testiranje:

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