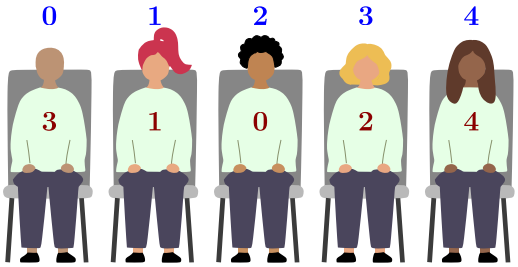
D. Istumajärjestys (seatingplan)

Tämän EGOI-tapahtuman päättäjäisseremoniaan osallistuu N tärkeää vierasta. Heidät kaikki pitää laittaa istumaan eturiviin hyvin tarkassa järjestyksessä, joka vastaa diplomatian protokollan kaikkia hienouksia. Oikean istumajärjestyksen selvittäminen vei Noemilta kaksi unetonta yötä.

Veronica on järjestämässä päättäjäisseremoniaa. Yksi hänen monista tehtävistään on varmistaa, että eturivin paikoilla on oikeat nimikyltit. On vain yksi pieni ongelma: Noemi ei koskaan kertonut hänelle oikeaa istumajärjestystä, eikä Noemia löydy enää mistään. Onneksi valokuvaajalla Dorkalla on sovellus, josta voisi olla hyötyä.

Dorkan piti valmistella kameransa, jotta hän voisi ottaa tiettynlaisia kuvia eturivin vieraista. Valmisteluja varten hänen piti tietää, kuinka leveä kukin kuva olisi, joten Noemi teki hänelle sovelluksen, joka antaa tarvittavat tiedot nopeasti. Veronica haluaa nyt käyttää sovellusta löytääkseen oikean istumajärjestyksen.

N tärkeää vierasta on numeroitu 0:sta $N-1$:teen. Eturivin paikat on myös numeroitu 0:sta $N-1$:teen, vasemmalta oikealle. Merkitään jokaiselle I:lle ($0 \leq I \leq N-1$) g_I:llä vierasta, jonka on tarkoitus istua paikalla I, ja s_I:llä paikkaa, jossa vieraan I on tarkoitus istua.



Kuva 1: Rivi, jossa on viisi vierasta. Tälle riville $g = [3, 1, 0, 2, 4]$ ja $s = [2, 1, 3, 0, 4]$.

Sovellus toimii seuraavasti:

- Dorka syöttää tasan kolmen eri vieraan numerot I, J ja K.
- Sovellus kertoo hänelle vähimmäismäärän vieraista, jotka ovat näkyvissä, jos kaikki kolme valittua vierasta ovat kuvassa. Formaalisti määriteltynä sovellus näyttää arvon $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Katso esimerkiksi Kuva 1 -kohdassa näkyvää tilannetta:

- Vieraat $I = 0$, $J = 2$ ja $K = 4$ ovat paikoilla $s_I = 2$, $s_J = 3$ ja $s_K = 4$. Jos Dorka valitsee heidät, sovellus näyttää arvon $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Toisin sanoen kapein kuva, joka sisältää vieraat 0, 2 ja 4, sisältää vain nämä kolme vierasta.

- Vieraat $I = 0$, $J = 4$ ja $K = 3$ ovat paikoilla $s_I = 2$, $s_J = 4$ ja $s_K = 0$. Jos Dorka valitsee heidät, sovellus näyttää arvon $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Toisin sanoen kuva, joka sisältää kolme annettua vierasta, on sisällettävä kaikki 5 vierasta.

Auta Veronicaa selvittämään oikea istumajärjestys Dorkan sovelluksen avulla. Tarkemmin sanottuna ohjelmasi tulee määrittää ja tulostaa järjestys $g_0, g_1, ..., g_{N-1}$. Oikeita vastauksia on aina tasan kaksi (joista toinen on toisen käänteinen), ja voit tulostaa kumman tahansa. Pisteesi riippuvat siitä, kuinka monta kyselyä sovellukselle ratkaisusi tekee.

Toteutus

⇨ Tämä on interaktiivinen tehtävä. Ohjelmasi kommunikoi pisteyttäjän kanssa käyttämällä vakiosyötettä ja -tulostetta alla kuvatun formaatin mukaisesti.

Ohjelmasi tulee aloittaa lukemalla yksi syöterivi, joka sisältää positiivisen kokonaisluvun T, eli testitapausten määrän, joka seuraa.

Jokaisen testitapauksen kohdalla ohjelmasi tulee aloittaa lukemalla yksi syöterivi, joka sisältää positiivisen kokonaisluvun N, eli paikkojen ja vieraiden määrän.

Tehdäksesi kyselyn, ohjelmasi tulee tulostaa rivi muodossa "? I J K", missä $0 \leq I, J, K \leq N-1$ ovat kolme **eri** lukua.

Kyselyn tekemisen jälkeen ohjelmasi tulee lukea yksi rivi, joka sisältää yhden positiivisen kokonaisluvun, eli kyselysi vastauksen.

Vastataksesi oikealla istumajärjestyksellä, ohjelmasi tulee tulostaa rivi muodossa "! g_0 … g_{N-1}".

Ratkaistuaan kaikki T testitapausta ohjelmasi tulee päättyä normaalisti.

Huomaa, että CMS-järjestelmässä ratkaisusi testaamiseen käytettävä virallinen pisteyttäjä saattaa olla **adaptiivinen**. Tämä tarkoittaa, että joidenkin testitapausten kohdalla vieraiden permutaatiota ei ole päätetty etukäteen. Sen sijaan pisteyttäjä saattaa muuttaa käyttäytymistään ohjelmasi tekemien kyselyiden perusteella.

Tulosteen tyhjennys (Flushing). Jos et käytä tarjottuja pohjia, varmista, että tyhjennät vakiotulosteen jokaisen rivin tulostamisen jälkeen, muuten ohjelmasi saatetaan tulkita vääräksi (*Not correct*). Pythonissa tämä tapahtuu automaattisesti, jos käytät `input()`-funktiota rivien lukemiseen, ja voit käyttää `print(..., flush=True)` pakottaaksesi tyhjennyksen. C++:ssa `cout << endl;` tyhjentää tulosteen rivinvaihdon lisäksi; jos käytät printf-funktiota, käytä `fflush(stdout)`.

Rajoitukset

- $1 \leq T \leq 10$.
- N on 5 (vain esimerkki), 8, 40 tai 2000.
- Jokaisessa testitapauksessa voit tehdä enintään 10 000 kyselyä.

Pisteytys

Ratkaisusi testataan useilla testitapauksilla, jotka on ryhmitelty osatehtäviin. Saadaksesi pisteet osatehtävästä, sinun on ratkaistava oikein kaikki sen sisältämät testit.

- **Osatehtävä 0 [0 pistettä]**: Esimerkki ($N = 5$).
- **Osatehtävä 1 [9 pistettä]**: $N = 8$.
- **Osatehtävä 2 [11 pistettä]**: $N = 2000$, ja vieraat 0 ja 1 istuvat vierekkäin.
- **Osatehtävä 3 [15 pistettä]**: $N = 40$.
- **Osatehtävä 4 [65 pistettä]**: $N = 2000$.

Osatehtävissä 1 ja 2 mikä tahansa ratkaisu, joka ratkaisee kaikki testitapaukset oikein, palkitaan täysillä pisteillä.

Osatehtävissä 3 ja 4 ratkaisusi on ratkaistava kaikki testitapaukset oikein saadaksesi pisteitä, ja pisteesi riippuvat arvosta Q_s, joka on suurin kyselyiden määrä, jonka ratkaisusi tarvitsi testitapauksen ratkaisemiseen. Olkoon $X_s = \max(1, Q_s/N)$. Osatehtävien 3 ja 4 pisteet lasketaan silloin seuraavasti:

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

score$_s$-arvo pyöristetään lähimpään kokonaislukuun osatehtävittäin, ja kokonaispisteesi on näiden summa. Saadaksesi täydet pisteet, sinun on ratkaistava osatehtävä 3 enintään 55 kyselyllä ja osatehtävä 4 enintään 2597 kyselyllä. Esimerkkiarvot Q_s:lle ja pisteille osatehtävissä 3 ja 4 on esitetty alla.

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

Esimerkit

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

(Yllä olevien esimerkkien ”Grader” on käännetty muualla tekstissä pisteyttäjäksi)

Selitys

Esimerkkisyöte sisältää yhden testitapauksen ($T = 1$), jossa on $N = 5$ vierasta. Tämän testitapauksen piilotettu vieraskokoonpano vastaa Kuva 1 -kohdan tilannetta.

Esimerkkiratkaisun ensimmäinen kysely on 0, 2, 4. Vastaus 3 tähän kyselyyn kertoo meille, että nämä vieraat istuvat jossain tuntemattomassa järjestyksessä kolmella vierekkäisellä paikalla.

Vastaus 3 toiseen kyselyyn kertoo meille saman vieraista 3, 0 ja 1.

Voimme nyt päätellä, että vieraan 0 on istuttava keskellä, vieraiden 2 ja 4 toisella puolella ja vieraiden 1 ja 3 toisella puolella.

Kolmannen kyselyn jälkeen voimme olla jo varmoja siitä, että vieraiden on istuttava joko järjestyksessä $[3, 1, 0, 2, 4]$ tai käänteisessä järjestyksessä $[4, 2, 0, 1, 3]$. Voimme tulostaa kumman tahansa järjestyksistä.

Koodipohjat ja arvioinnin yksityiskohdat CMS-järjestelmässä

Suosittelemme vahvasti käyttämään tarjottua koodipohjia C++:salle sekä Pythonille. Ne tarkistavat, että kommunikointi pisteyttäjän kanssa onnistui sekä lopettavat ohjelman siististi sen epäonnistuessa.

Jos et käytä annettuja pohjia, CMS voi näyttää virheellisen ilmoituksen tapauksissa, joissa vastauksesi on väärin. Esimerkiksi sen sijaan, että saisit ilmoituksen ”Output isn’t correct”, voisit saada ilmoituksen ”Execution killed by signal” tai ”Execution timed out (wall clock limit exceeded)”.

Suosittelemme myös vahvasti käyttämään tarjottua testaustyökalua (katso alta) ratkaisusi testaamiseen paikallisesti ennen sen lähettämistä. Testaustyökalu tarkistaa ratkaisujesi tulosteet ja ilmoittaa kelpaamattomista kyselyistä.

Testaustyökalu

Ratkaisusi testaamisen helpottamiseksi tarjoamme yksinkertaisen työkalun, jonka voit ladata CMS-järjestelmästä. Työkalun käyttö on vapaaehtoista. Huomaa, että CMS:n virallinen pisteyttäjä on erilainen kuin testaustyökalu.

Työkalun käyttämiseen tarvitset syötetiedoston. Voit käyttää mukana tulevaa esimerkkisyötettä `seatingplan.input0.txt` tai tehdä oman. Syötetiedoston tulee alkaa rivillä, jossa on testitapausten määrä T, ja sen jälkeen jokaiselle testitapaukselle tulee olla kaksi riviä: yksi rivi, jossa on luku N, ja yksi rivi, jossa ovat luvut $g_0, g_1, ..., g_{N-1}$.

Python-ohjelmille, esimerkiksi `seatingplan.py` (tavallisesti ajetaan komennolla `pypy3 seatingplan.py`), aja testaustyökalun seuraavasti:

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

C++-ohjelmille, käännä ensin ratkaisusi:

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

ja aja sitten testaustyökalu:

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