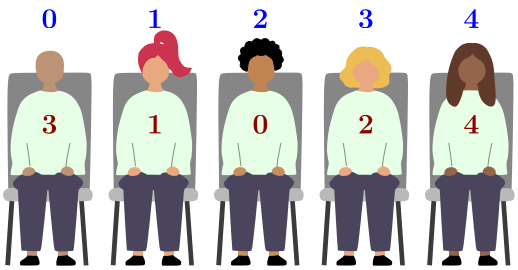
D. Sėdėjimo planas (seatingplan)

Šios EGOI uždarymo ceremonijoje dalyvaus N svarbių svečių. Visus juos reikia susodinti pirmoje eilėje pagal labai konkrečią tvarką, atitinkančią visus diplomatinio protokolo niuansus. Teisingos sėdėjimo tvarkos sudarymas Noemi kainavo dvi bemieges naktis.

Veronika yra atsakinga už uždarymo ceremoniją. Viena iš daugelio jos pareigų – užtikrinti, kad ant pirmos eilės kėdžių būtų sudėliotos teisingos vardų kortelės. Yra tik viena maža problemėlė: Noemi Veronikai taip ir nepasakė sudarytos teisingos sėdėjimo tvarkos, o pačios Noemi dabar niekur nepavyksta rasti. Laimei, fotografė Dorka turi programėlę, kuri galėtų praversti.

Dorka turėjo paruošti savo fotoaparatus taip, kad ji galėtų padaryti kelias specifines pirmos eilės svečių nuotraukas. Tam, kad pavyktų paruošti fotoaparatus, Dorkai reikėjo sužinoti, kokio pločio turėtų būti kiekviena nuotrauka. Dėl šios priežasties Noemi jai sukūrė programėlę, kuri greitai išveda reikiamą informaciją. Dabar Veronika nori pasinaudoti šia programėle, kad išsiaiškintų teisingą sėdėjimo tvarką.

N svarbių svečių yra sunumeruoti nuo 0 iki $N-1$. Kėdės pirmoje eilėje taip pat sunumeruotos nuo 0 iki $N-1$, numeruojant iš kairės į dešinę. Kiekvienam I ($0 \le I \le N-1$), tegu g_I žymi svečią, kuris turi sėdėti kėdėje I, o s_I žymi kėdę, kurioje turi sėdėti svečias I.



Pav. 1: Eilė su penkiais svečiais. Šiai eilei $g = [3, 1, 0, 2, 4]$ ir $s = [2, 1, 3, 0, 4]$.

Programėlė veikia taip:

- Dorka įveda lygiai trijų skirtingų svečių numerius I, J, K.
- Programėlė jai parodo, koks mažiausias svečių skaičius bus matomas, jei visi trys pasirinkti svečiai bus nuotraukoje. Pagal apibrėžimą programėlė parodys reikšmę $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Pavyzdžiui, pažiūrėk į situaciją, pavaizduotą Pav. 1:

- Svečiai $I = 0$, $J = 2$ ir $K = 4$ sėdi kėdėse $s_I = 2$, $s_J = 3$ ir $s_K = 4$. Jei Dorka juos pasirenka, programėlė parodys reikšmę $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Kitaip tariant, siauriausioje nuotraukoje, kurioje telpa svečiai 0, 2 ir 4, yra tik šie trys svečiai.

- Svečiai $I = 0$, $J = 4$ ir $K = 3$ sėdi kėdėse $s_I = 2$, $s_J = 4$ ir $s_K = 0$. Jei Dorka juos pasirenka, programėlė parodys reikšmę $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Kitaip tariant, nuotraukoje, kurioje yra trys nurodyti svečiai, privalo tilpti visi 5 svečiai.

Padėk Veronikai nustatyti teisingą sėdėjimo tvarką naudojantis Dorkos programėle. Konkrečiau, tavo programa turėtų nustatyti ir išvesti seką $g_0, g_1, ..., g_{N-1}$. Visada yra lygiai du teisingi atsakymai (vienas yra kito atvirkštinis variantas), ir gali išvesti bet kurį iš jų. Tavo taškai priklausys nuo to, kiek užklausų programėlei pateiks tavo sprendimas.

Implementacija

⇨ Tai interaktyvus uždavinys. Tavo programa naudos standartinę įvestį ir išvestį (standard input/output), kad bendrautų su vertinimo programa (angl. grader) žemiau nurodytu formatu.

Tavo programa pirmiausia turėtų nuskaityti vieną įvesties eilutę, kurioje yra teigiamas sveikasis skaičius T – toliau sekančių testų skaičius.

Kiekvienam testui tavo programa pirmiausia turėtų nuskaityti vieną įvesties eilutę, kurioje yra teigiamas sveikasis skaičius N – kėdžių skaičius, kuris taip pat yra ir svečių skaičius.

Norėdama pateikti užklausą, tavo programa turėtų išvesti eilutę formatu „? I J K“, kur $0 \leq I, J, K \leq N-1$ yra trys **skirtingi** skaičiai.

Pateikusi užklausą, tavo programa turėtų nuskaityti vieną eilutę su vienu teigiamu sveikuoju skaičiumi – atsakymu į tavo užklausą.

Norėdama pateikti atsakymą su teisinga sėdėjimo tvarka, tavo programa turėtų išvesti eilutę formatu „! g_0 ⋯ g_{N-1}“.

Išsprendusi visus T testų, tavo programa turėtų sėkmingai baigti darbą (angl. terminate normally).

Atkreipk dėmesį, kad oficiali testavimo sistema CMS aplinkoje, naudojama testuoti tavo sprendimą, gali būti **adaptyvi**. Tai reiškia, kad kai kuriuose testuose svečių išsidėstymas nėra nuspręstas iš anksto. Vietoje to, testavimo sistema gali nuspręsti, kurį iš likusių galimų išsidėstymų naudoti, priklausomai nuo to, kokias užklausas jau buvo pateikusi tavo programa.

Flushing procedūra. Jei nenaudoji pateiktų šablonų, būtinai atlik procedūrą **flush** po kiekvienos išvestos eilutės, kitaip tavo programa gali būti įvertinta kaip *Netinkamas (Not correct)*. Python kalboje **flush** procedūra vyksta automatiškai, jei eilutėms skaityti naudoji `input()` ir gali naudoti `print(..., flush=True)`, kad priverstinai atliktum **flush** procedūrą. C++ kalboje `cout << endl;` ne tik išveda naują eilutę, bet ir atlieka **flush** procedūrą; bet jei naudoji `printf`, tai naudoki `fflush(stdout)`.

Apribojimai

- $1 \leq T \leq 10$.
- N bus 5 (tik pavyzdyje), 8, 40 arba 2000.
- Kiekviename teste gali pateikti ne daugiau kaip 10 000 užklausų.

Vertinimas

Tavo programa bus tikrinama su keliais testavimo atvejais, suskirstytais į testų grupes. Norėdama gauti taškus už testų grupę, turi teisingai išspręsti visus jos testavimo atvejus.

- **0-a testų grupė [0 taškų]**: Pavyzdys ($N = 5$).
- **1-a testų grupė [9 taškai]**: $N = 8$.
- **2-a testų grupė [11 taškų]**: $N = 2000$, o svečiai 0 ir 1 sėdi vienas šalia kito.
- **3-a testų grupė [15 taškų]**: $N = 40$.
- **4-a testų grupė [65 taškai]**: $N = 2000$.

1 ir 2 testų grupėms bet koks sprendimas, teisingai išsprendęs visus testavimo atvejus, bus įvertintas pilnu taškų skaičiumi.

3 ir 4 testų grupėse tavo sprendimas turi teisingai išspręsti visus testavimo atvejus, kad gautum bent kiek taškų, o tavo rezultatas priklausys nuo Q_s – didžiausio užklausų skaičiaus, kurio prireikė vienam testui išspręsti. Pažymėkime $X_s = \max(1, Q_s/N)$. Tuomet 3 ir 4 testų grupių taškai skaičiuojami taip:

$$3_{\text{įvertinimas}} = \min\left(15, 3 + \frac{19}{X_s^{1.5}}\right), \qquad 4_{\text{įvertinimas}} = \min\left(65, 3 + \frac{91}{X_s^{1.5}}\right)$$

$s_{\text{įvertinimas}}$ reikšmė apvalinama iki artimiausio sveikojo skaičiaus kiekvienoje testų grupėje, o tavo galutinis rezultatas yra jų suma. Norėdama surinkti visus taškus, 3 testų grupę turi išspręsti panaudodama ne daugiau kaip 55 užklausas, o 4 testų grupę – ne daugiau kaip 2597 užklausas.

Žemiau pateikiamos pavyzdinės Q_s reikšmės ir įvertinimai 3 ir 4 testų grupėms.

Q_s	55	56	60	70	80	100	150	10000
įvertinimas$_3$	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
įvertinimas$_4$	65	58	53	35	26	21	14	11

Pavyzdžiai

Vertinimo programa	Atsakymas
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Paaiškinimas

Pavyzdiniuose pradiniuose duomenyse yra vienas testas ($T = 1$) su $N = 5$ svečiais. Paslėpta svečių konfigūracija šiame teste atitinka Pav. 1.

Pirmoji užklausa, kurią pateikė pavyzdinis sprendimas, yra 0, 2, 4. Šios užklausos atsakymas, kuris yra 3, nurodo, kad šie svečiai sėdi (kažkokia nežinoma tvarka) trijose iš eilės einančiose kėdėse greta vienas kito.

Antrosios užklausos atsakymas, kuris yra 3, nurodo tą patį apie svečius 3, 0 ir 1.

Dabar galime padaryti išvadą, kad 0-as svečias turi sėdėti viduryje, su 2-u ir 4-u svečiais vienoje pusėje bei 1-u ir 3-iu svečiais kitoje pusėje.

Po trečiosios užklausos jau galime būti tikri, kad svečiai sėdi arba tvarka $[3, 1, 0, 2, 4]$, arba atvirkštine tvarka $[4, 2, 0, 1, 3]$. Galime išvesti bet kurią iš šių tvarkų.

Kodo šablonai ir vertinimo CMS sistemoje detalės

Primygtinai rekomenduojame naudoti pateiktus C++ ir Python kodo šablonus. Jie patikrina, ar ryšys su vertinimo programa sėkmingas, ir nesėkmingo ryšio atveju gražiai užbaigia darbą.

Jei nenaudosi pateiktų šablonų, atvejais, kai tavo sprendimas neteisingas, CMS gali pateikti klaidingą verdiktą. Pavyzdžiui, vietoj „Output isn't correct“ gali gauti „Execution killed by signal“ arba „Execution timed out (wall clock limit exceeded)“.

Taip pat rekomenduojame naudoti testavimo įrankį (žr. žemiau) savo sprendimo testavimui lokaliai prieš pateikiant. Testavimo įrankis tikrina tavo sprendimo išvedimą ir praneša, jei naudojamos negalimos užklausos.

Testavimo įrankis

Kad būtų lengviau išbandyti sprendimą, pateikiame paprastą įrankį, kurį gali atsisiųsti iš CMS. Įrankiu naudotis neprivaloma. Atkreipk dėmesį, kad oficiali vertinimo programa CMS sistemoje skiriasi nuo šio testavimo įrankio.

Norint naudoti įrankį, tau reikės pradinių duomenų failo. Gali naudoti pateiktą pavyzdį `seatingplan.input0.txt` arba susikurti savo. Pradinių duomenų failas turėtų prasidėti eilute, kurioje yra testų skaičius T, o toliau kiekvienam testui turėtų būti dvi eilutės: viena su skaičiumi N, o kita su skaičiais $g_0, g_1, ..., g_{N-1}$.

Python programai, tarkime `seatingplan.py` (paprastai paleidžiamai su `pypy3 seatingplan.py`), testavimo įrankį paleisk taip:

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

C++ programoms pirmiausia sukompiliuok savo sprendimą:

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

ir tada paleisk testavimo įrankį:

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