

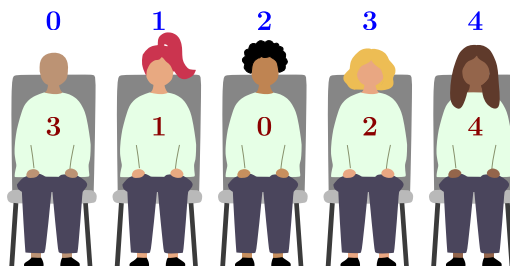
## D. Ültetési rend (seatingplan)

Az idei EGOI záróünnepségén  $N$  fontos vendég vesz részt. Mindannyiukat az első sorba kell ültetni egy nagyon meghatározott sorrendben, amely minden diplomáciai protokollnak megfelel. Noéminek két álmatlan éjszakájába telt meghatározni a helyes ültetési sorrendet.

Veronika felügyeli a záróünnepséget. Számos feladata közül az egyik az, hogy megbizonyosodjon arról, hogy az első sorban lévő székekre a megfelelő névtáblák kerülnek. Csak egy apró probléma van: Noémi sosem mondta el neki a helyes ültetési sorrendet, és most sehol sem találják Noémit. Szerencsére Dorka, a fotós rendelkezik egy alkalmazással, ami hasznos lehet.

Dorkának elő kellett készítenie a kameráit, hogy fényképeket készíthessen az első sorban ülő vendégekről. A kamera beállításához tudnia kellett, milyen szélesek lesznek az egyes fotók, ezért Noémi készített neki egy alkalmazást, amely gyorsan kiírja a szükséges információkat. Veronika most ezt az alkalmazást szeretné használni a helyes ülésrend megtalálásához.

Az  $N$  fontos vendéget 0-tól  $N - 1$ -ig számozták. Az első sorban lévő székeket szintén 0-tól  $N - 1$ -ig számozták, balról jobbra. Minden  $i$  ( $0 \leq i \leq N - 1$ ) esetén jelölje  $g_i$  azt a vendéget, akinek az  $i$ -edik székre kell ülnie, és jelölje  $s_i$  azt a széket, amelyre az  $i$ -edik vendégnek kell ülnie.



Ábra 1: Egy sor öt vendéggel. Ebben a sorban  $g = [3, 1, 0, 2, 4]$  és  $s = [2, 1, 3, 0, 4]$ .

Az alkalmazás a következőképpen működik:

- Dorka megadja három különböző vendég  $i$ ,  $j$ ,  $k$  sorszámát.
- Az alkalmazás megmondja neki, hány vendég látszik minimálisan, ha mindhárom kiválasztott vendég rajta van a képen. Formálisan az alkalmazás a  $\max(s_i, s_j, s_k) - \min(s_i, s_j, s_k) + 1$  értéket jeleníti meg.

Például nézzük az 1. ábrán látható helyzetet:

- A 0, 2 és 4 sorszámú vendégek a 2, 3 és 4 székeken ülnek. Ha Lisa őket választja, az alkalmazás a  $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$  értéket írja ki.

Más szóval, a legkeskenyebb fotó, amely tartalmazza a 0, 2 és 4 vendéget, az csak ezt a három vendéget tartalmazza.

- A 0, 4 és 3 sorszámú vendégek a 2, 4 és 0 székeken ülnek. Ha Lisa őket választja, az alkalmazás a  $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$  értéket írja ki.

Más szóval, egy fotó, amely tartalmazza a három adott vendéget, annak mind az 5 vendéget tartalmaznia kell.

Segíts Veronikának meghatározni a helyes ültetési sorrendet a Dorka által használt alkalmazás segítségével. Pontosabban, a programodnak meg kell határoznia és ki kell írnia a  $g_0, g_1, \dots, g_{N-1}$  sorozatot.

Mindig pontosan két helyes válasz létezik (az egyik a másik fordítottja), és bármelyiket megadhatod. A pontszámod attól függ, hány lekérdezést hajt végre a megoldásod az alkalmazásban.

## Implementáció



Ez egy interaktív feladat. A programod a standard bemenetet és kimenetet használja a kommunikációhoz egy ellenőrző programmal az alábbi formátumban.

A programodnak úgy kell indulnia, hogy beolvasson egy sort, amely tartalmazza a  $T$  pozitív egész számot, a tesztesetek számát.

Minden tesztesethez a programodnak egy pozitív egész számot  $N$ -et kell beolvasnia, ami a székek száma és egyben a vendégek száma is.

Lekérdezéshez a programodnak egy ilyen formátumú sort kell kiírnia: „?  $i\ j\ k$ ”, ahol  $0 \leq i, j, k \leq N - 1$  három **különböző** szám.

Lekérdezés után a programodnak be kell olvasnia egy sort, amely egy pozitív egész számot tartalmaz, ez a lekérdezésedre a válasz.

A helyes ültetési sorrend megadásához a programodnak egy ilyen formátumú sort kell kiírnia: „!  $g_0 \dots g_{N-1}$ ”.

Az összes  $T$  teszteset megoldása után a programodnak szabályosan be kell fejeződnie.

Ne feledd, hogy a CMS-ben használt hivatalos ellenőrző program **adaptív** lehet. Ez azt jelenti, hogy egyes teszteseteknél a vendégek permutációja nincs előre meghatározva. Ehelyett az ellenőrző program a programod által feltett kérdésektől függően változtathatja a viselkedését.

**Pufferezés (Flushing).** Ha nem a megadott sablonokat használod, győződj meg róla, hogy minden sor kiírása után üríted (flush) a standard kimenetet, különben a programod **Helytelen** minősítést kaphat. C++-ban a `cout << endl;` a sortörés mellett üríti is a puffert; ha `printf`-et használsz, használd az `fflush(stdout)` parancsot. Pythonban ez automatikus, ha az `input()` függvényt használod olvasásra és használhatod a `print(..., flush=True)` utasítást az ürítés biztos elvégzéséhez.

## Korlátok

- $1 \leq T \leq 10$ .
- $N$  értéke 5 (csak példa), 8, 40 vagy 2000 lehet.
- Minden tesztesethez legfeljebb 10 000 lekérdezést hajthatsz végre.

## Pontozás

A programodat több, részfeladatokba csoportosított teszteseten teszteljük. Ahhoz, hogy megkapd egy részfeladat pontszámát, az összes benne lévő tesztesetet helyesen kell megoldanod.

- **0. Részfeladat [ 0 pont]:** Példa ( $N = 5$ ).
- **1. Részfeladat [ 9 pont]:**  $N = 8$ .
- **2. Részfeladat [11 pont]:**  $N = 2000$ , és a 0 és 1 sorszámú vendégek egymás mellett ülnek.
- **3. Részfeladat [15 pont]:**  $N = 40$ .
- **4. Részfeladat [65 pont]:**  $N = 2000$ .

Az 1. és 2. részfeladatnál bármilyen megoldás, amely az összes tesztesetet helyesen megoldja, megkapja az összes pontot.

A 3. és 4. részfeladatnál a megoldásodnak minden tesztesetet helyesen kell megoldania, hogy pontot szerezhess, és a pontszámod a  $Q_s$ -től függ, ami a legnagyobb lekérdezésszám, amit a megoldásodnak egy teszteset megoldásához végre kellett hajtania. Legyen  $X_s = \max(1, Q_s/N)$ . A 3. és 4. részfeladat score-ja (kapott pontszáma):

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

A  $\text{score}_s$  értékét részfeladatonként kerekítjük a legközelebbi egész számra és a teljes pontszámod ezek összege. A maximális pontszám eléréséhez a 3. részfeladatot legfeljebb 55 lekérdezésből, a 4. részfeladatot pedig legfeljebb 2597 lekérdezésből kell megoldanod. Példaértékek a kérdések ( $Q_s$ ) számára és az arra kapott pontszámra a 3. és a 4. részfeladatban:

$Q_s$	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

## Példák

Értékelő	Megoldás
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

## Magyarázat

A példa bemenet egy tesztesetet ( $T = 1$ ) tartalmaz  $N = 5$  vendéggel. Ebben a tesztesetben a vendégek rejtett elrendezése megfelel az 1. ábrának.

Az első, a példamegoldás által elvégzett lekérdezés: 0, 2, 4. A lekérdezésre adott 3-as válasz azt jelzi nekünk, hogy ezek a vendégek valamilyen ismeretlen sorrendben három egymás melletti széken ülnek.

A második lekérdezésre adott 3-as válasz ugyanezt mondja el a 3, 0 és 1 sorszámú vendégekről.

Ebből kikövetkeztethetjük, hogy a 0. vendégnek középen kell ülnie, az egyik oldalon a 2. és 4. vendéggel, a másikon pedig az 1. és 3. vendéggel.

A harmadik lekérdezés után már biztosak lehetünk benne, hogy a vendégek vagy  $[3, 1, 0, 2, 4]$  sorrendben, vagy a fordított  $[4, 2, 0, 1, 3]$  sorrendben ülnek. Bármelyik sorrendet megadhatjuk válaszként.

## Kódsablonok és kiértékelési részletek a CMS-ben

Erősen javasoljuk a mellékelt C++ és Python kódsablonok használatát. Ezek ellenőrzik, hogy a kommunikáció az értékelővel (graderrel) sikeres volt-e, és szabályosan leállnak, ha nem.

A megoldásoddal kommunikáló értékelő az első észlelt hibát jelenti, majd leáll. Ha nem a mellékelt sablonokat használod, azokban az esetekben, amikor a megoldásod nem jó, a CMS rossz értékelést ír ki. Például a CMS rossz értékelést ír ki. Például az „Output isn’t correct” üzenet helyett kaphatsz „Execution killed by signal” vagy „Execution timed out (wall clock limit exceeded)” üzenetet is.

Javasoljuk, hogy beküldés előtt a mellékelt tesztelőeszközzel (lásd lentebb) helyben teszteld a megoldásodat. A tesztelőeszköz ellenőrzi a megoldásod kimenetét és jelzi az érvénytelen lekérdezéseket.

## Tesztelőeszköz

A megoldásod tesztelésének megkönnyítésére egy egyszerű eszközt biztosítunk, amelyet a CMS-ből tölthetsz le. Az eszköz használata nem kötelező. Ne feledd, hogy a CMS-en lévő hivatalos ellenőrző program eltér a CMS tesztelő eszközétől.

Az eszköz használatához egy bemeneti fájlra van szükséged. Használhatod a mellékelt `seatingplan.input0.txt` példa bemenetet, vagy készíthetsz sajátot. A bemeneti fájlnek egy olyan sorral kell kezdődnie, amely tartalmazza a  $T$  tesztesetek számát, majd tesztesetenként két sort kell tartalmaznia: az egyik sorban az  $N$  számot, a másikban pedig a  $g_0, g_1, \dots, g_{N-1}$  számokat.

Python programokhoz (általában `pypy3 seatingplan.py` paranccsal futtatva) az alábbi módon futtasd a tesztelő eszközt:

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

C++ programokhoz először fordítsd le a megoldásodat:

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

majd futtasd a tesztelő eszközt:

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