D. Распоред на седење (seatingplan)

На церемонијата за затворање на оваа EGOI ќе присуствуваат N важни гости. Сите тие треба да седат во првиот ред по многу специфичен редослед кој одговара на сите нијанси на дипломатскиот протокол. Одредувањето на правилниот редослед на седење ѝ одземал на Ноеми две непроспиени ноќи.

Вероника ја надгледува церемонијата за затворање. Една од нејзините многубројни одговорности е да се осигура дека седиштата во првиот ред ги имаат точните ознаки со имиња. Има само еден мал проблем: Ноеми никогаш не ѝ го кажала правилниот редослед на седење, а сега Ноеми не може да се најде никаде. За среќа, фотографката Дорка има апликација која би можела да биде корисна.

Дорка морала да ги подготви своите камери за да може да направи некои специфични фотографии од гостите во првиот ред. За поставувањето, таа требала да знае колку ќе биде широка секоја фотографија, па Ноеми ѝ направила апликација која брзо ѝ ги дава потребните информации. Вероника сега сака да ја искористи апликацијата за да го најде правилниот распоред на седиштата.

N-те важни гости се нумерирани од 0 до $N-1$. Седиштата во првиот ред исто така се нумерирани од 0 до $N-1$, од лево кон десно. За секое I $(0 \leq I \leq N-1)$, нека g_I го означува гостинот кој треба да седи на седиштето I, а s_I нека го означува седиштето на кое треба да седи гостинот I.

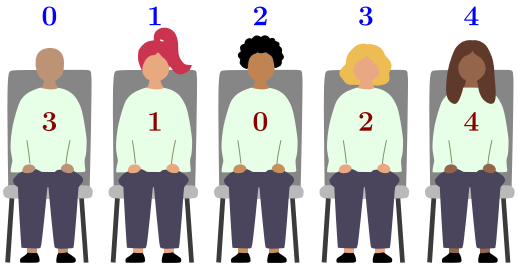

Figure 1: Ред со пет гости. За овој ред, $g = [3, 1, 0, 2, 4]$ и $s = [2, 1, 3, 0, 4]$.

Апликацијата работи на следниов начин:

- Дорка ги внесува броевите I, J, K на точно тројца различни гости.
- Апликацијата ѝ кажува кој е минималниот број на гости кои ќе бидат видливи ако сите тројца избрани гости се на фотографијата. Формално, апликацијата ќе ја прикаже вредноста $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

На пример, види ја ситуацијата прикажана на Figure 1:

- Гостите $I = 0$, $J = 2$ и $K = 4$ се на седиштата $s_I = 2$, $s_J = 3$ и $s_K = 4$. Ако Дорка ги избере нив, апликацијата ќе ја прикаже вредноста $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Со други зборови, најтесната фотографија која ги содржи гостите 0, 2 и 4 ги има само овие тројца гости.

- Гостите $I = 0$, $J = 4$ и $K = 3$ се на седиштата $s_I = 2$, $s_J = 4$ и $s_K = 0$. Ако Дорка ги избере нив, апликацијата ќе ја прикаже вредноста $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Со други зборови, фотографија која ги содржи дадените тројца гости мора да ги содржи сите 5 гости.

Помогни ѝ на Вероника да го одреди правилниот редослед на седење користејќи ја апликацијата на Дорка. Поконкретно, твојата програма треба да ја одреди и испечати низата $g_0, g_1, ..., g_{N-1}$. Секогаш има точно два точни одговори (едниот е обратен од другиот), и ти можеш да го испечатиш било кој од нив. Твоите поени ќе зависат од бројот на прашања до апликацијата што ќе ги постави твоето решение.

Implementation

> ⇨ Ова е интерактивна задача. Твојата програма ќе користи стандарден влез и излез за комуникација со оценувачот (grader) во форматот опишан подолу.

Твојата програма треба да започне со читање на една линија од влезот која содржи позитивен цел број T, бројот на тест примери кои следуваат.

За секој тест пример, твојата програма треба да започне со читање на една линија од влезот која содржи позитивен цел број N, бројот на седишта, што е исто така и бројот на гости.

За да поставиш прашање (query), твојата програма треба да испечати линија во формат "? I J K", каде $0 \leq I, J, K \leq N-1$ се три **различни** броеви.

Откако ќе поставиш прашање, твојата програма треба да прочита една линија која содржи еден позитивен цел број, одговорот на твоето прашање.

За да одговориш со правилниот редослед на седење, твојата програма треба да испечати линија во формат "! g_0 … g_{N-1}".

Откако ќе ги реши сите T тест примери, твојата програма треба нормално да заврши.

Да забележиме дека официјалниот оценувач што се користи во CMS за тестирање на твоето решение може да биде **адаптивен**. Тоа значи дека за некои тест примери, пермутацијата на гостите не е однапред одредена. Наместо тоа, оценувачот може да одлучи која од преостанатите пермутации ќе ја користи, во зависност од прашањата што веќе биле поставени од твојата програма.

Чистење на баферот (Flushing). Ако не ги користиш дадените шаблони, осигури се дека го чистиш стандардниот излез откако ќе ја испечатиш секоја линија, инаку твојата програма може да биде оценета како *Неточна (Not correct)*. Во Python, ова се случува автоматски ако користиш `input()` за читање линии, а можеш да користиш `print(..., flush=True)` за присилно чистење. Во C++, `cout << endl;` го чисти баферот покрај тоа што печати нов ред; ако користиш `printf`, користи `fflush(stdout)`.

Constraints

- $1 \leq T \leq 10$.
- N ќе биде 5 (само за примерот), 8, 40, или 2000.
- За секој тест пример, можеш да поставиш најмногу 10 000 прашања.

Scoring

Твојата програма ќе биде тестирана на повеќе тест примери групирани во подзадачи. За да ги добиеш поените за дадена подзадача, мораш точно да ги решиш сите тестови што таа ги содржи.

- **Subtask 0** [**0 points**]: Пример ($N = 5$).
- **Subtask 1** [**9 points**]: $N = 8$.
- **Subtask 2** [**11 points**]: $N = 2000$, и гостите 0 и 1 седат еден до друг.
- **Subtask 3** [**15 points**]: $N = 40$.
- **Subtask 4** [**65 points**]: $N = 2000$.

За подзадачите 1 и 2, секое решение кое точно ќе ги реши сите тест примери ќе ги добие сите поени.

За подзадачите 3 и 4, твоето решение мора точно да ги реши сите тест примери за да добие поени, а твоите поени ќе зависат од Q_s, најголемиот број на прашања што му биле потребни на твоето решение да ги постави за да реши еден тест пример. Нека $X_s = \max(1, Q_s/N)$. Поените за подзадачите 3 и 4 потоа се пресметуваат на следниот начин:

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

Вредноста на score_s се заокружува на најблискиот цел број по подзадача, а твоите вкупни поени се сумата од овие. За да добиеш целосни поени, треба да ја решиш подзадачата 3 во најмногу 55 прашања и подзадачата 4 во најмногу 2597 прашања. Примери за вредности на Q_s и поени за подзадачите 3 и 4 се прикажани подолу.

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

Examples

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

Explanation

Примерот содржи еден тест пример ($T = 1$) со $N = 5$ гости. Скриената конфигурација на гости во овој тест пример одговара на Figure 1.

Првото прашање поставено од решението за примерот е 0, 2, 4. Одговорот 3 на ова прашање ни кажува дека овие гости седат, во некој непознат редослед, на три последователни седишта еден до друг.

Одговорот 3 на второто прашање ни го кажува истото за гостите 3, 0 и 1.

Сега можеме да заклучиме дека гостинот 0 мора да седи во средина, со гостите 2 и 4 на едната страна и гостите 1 и 3 на другата страна.

По третото прашање, веќе можеме да бидеме сигурни дека гостите мора да седат или по редослед $[3, 1, 0, 2, 4]$ или во обратен редослед $[4, 2, 0, 1, 3]$. Можеме да испечатиме кој било од редоследите.

Кодни шаблони и детали за оценување во CMS

Силно препорачуваме да ги користиш обезбедените кодни шаблони за C++ и Python. Тие проверуваат дали комуникацијата со оценувачот била успешна и терминираат на соодветен начин ако не била.

Ако не ги користиш обезбедените шаблони, во случаи кога твоето решение е неточно, CMS може да прикаже погрешна пресуда. На пример, наместо „Output isn't correct“ може да добиеш „Execution killed by signal“ или „Execution timed out (wall clock limit exceeded)“.

Исто така, препорачуваме алатка за тестирање (види подолу) за да го тестираш твоето решение локално пред да го пратиш. Алатката за тестирање ги проверува излезите на твоето решение и пријавува користење невалидни прашања.

Алатка за тестирање

За да го олесниме тестирањето на твоето решение, обезбедуваме едноставна алатка која можеш да ја преземеш од CMS. Користењето на алатката не е задолжително. Да забележиме дека официјалниот оценувач на CMS е различен од алатката за тестирање.

За да ја користиш алатката, потребна ти е влезна датотека. Можеш да го искористиш дадениот пример за влез `seatingplan.input0.txt` или да направиш свој. Влезната датотека треба да започне со линија која го содржи бројот T на тест примери, а потоа треба да има по две линии за секој тест пример: една линија со бројот N и потоа една линија со броевите $g_0, g_1, ..., g_{N-1}$.

За Python програми, на пример `seatingplan.py` (обично се извршува како `pypy3 seatingplan.py`) стартувај ја алатката за тестирање на следниов начин:

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

За C++ програми, прво компајлирај го твоето решение:

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

и потоа стартувај ја алатката за тестирање:

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