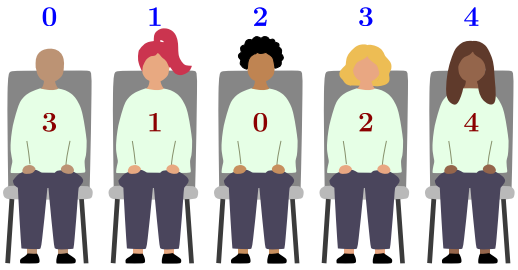
D. Penempatan Duduk (seatingplan)

Penutupan EGOI tahun ini akan dihadiri oleh N orang tamu. Mereka semua harus duduk di baris depan dengan urutan yang spesifik.

Veronica bertugas untuk mengawasi acara tersebut dan bertanggung jawab untuk memastikan kursi baris depan memiliki label yang benar. Namun, Veronica tidak tahu urutan tempat duduk yang benar. Untungnya Dorka, sang kameramen, memiliki aplikasi yang dapat membantu.

Dorka menyiapkan kameranya untuk mengambil foto spesifik dari para tamu tersebut. Di pengaturan kameranya, Dorka perlu mengetahui seberapa lebar setiap foto ketika mengambil foto spesifik dari para tamu. Aplikasi yang dimiliki Dorka mengeluarkan informasi yang dibutuhkan Veronica untuk menemukan penempatan tempat duduk yang benar.

N tamu tersebut dinomori dari 0 hingga $N-1$. Kursi di baris depan juga dinomori dari 0 hingga $N-1$ (dari kiri ke kanan). Untuk setiap I $(0 \leq I \leq N-1)$, g_I (**guest**) menunjukkan tamu yang seharusnya duduk di kursi I, dan s_I (**seat**) menunjukkan kursi tempat tamu I seharusnya duduk.



Figur 1 : Barisan dengan lima tamu. Untuk baris ini, $g = [3, 1, 0, 2, 4]$ dan $s = [2, 1, 3, 0, 4]$.

Aplikasinya bekerja sebagai berikut:

- Dorka memasukkan nomor I, J, K dari tiga tamu yang berbeda.
- Aplikasi mengembalikan jumlah minimum tamu yang akan terlihat jika ketiga tamu yang dipilih berada dalam sebuah foto. Secara formal, $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Lihat Figur 1 sebagai contoh:

- Tamu $I = 0$, $J = 2$, dan $K = 4$ berada di kursi $s_I = 2$, $s_J = 3$, dan $s_K = 4$. Jika Dorka memasukkan mereka, aplikasi akan mengembalikan $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Dengan kata lain, foto tersempit yang berisi tamu 0, 2, dan 4 hanya berisi ketiga tamu tersebut.

- Tamu $I = 0$, $J = 4$, dan $K = 3$ berada di kursi $s_I = 2$, $s_J = 4$, dan $s_K = 0$. Jika Dorka memasukkan mereka, aplikasi akan mengembalikan $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Dengan kata lain, foto yang berisi ketiga tamu tersebut pasti berisi semua tamu.

Bantulah Veronica menentukan urutan tempat duduk yang benar menggunakan aplikasi milik Dorka. Secara spesifik, program Anda harus mengeluarkan urutan $g_0, g_1, ..., g_{N-1}$. Akan selalu ada tepat dua jawaban yang benar (yang satu pasti kebalikan dari yang lainnya), dan Anda boleh mengeluarkan salah satu dari mereka. Poin Anda bergantung dengan jumlah *query* yang diberikan ke aplikasi.

Implementasi

⇨ Masalah ini bersifat interaktif. Program Anda akan menggunakan *standard input* dan *output* untuk berkomunikasi dengan grader dalam format yang dijelaskan di bawah.

Program Anda harus dimulai dengan membaca sebuah bilangan bulat positif T, yaitu jumlah kasus uji.

Untuk setiap kasus uji, program Anda harus membaca sebuah bilangan bulat positif N, yaitu jumlah kursi dan tamu.

Untuk membuat sebuah query, program Anda harus mengeluarkan satu baris dengan format "? I J K", di mana $0 \leq I, J, K \leq N-1$ adalah tiga angka yang **berbeda**.

Setelah itu, program Anda harus membaca sebuah bilangan bulat positif (jawaban dari query Anda).

Program Anda harus mengeluarkan satu baris dengan format "! g_0 … g_{N-1}" untuk menjawab.

Setelah menyelesaikan semua T kasus uji, program Anda harus berhenti secara normal.

Perhatikan bahwa grader yang digunakan di CMS untuk menilai solusi Anda mungkin bersifat **adaptif**. Artinya, permutasi tamu untuk beberapa kasus uji bisa jadi belum ditentukan. Grader mungkin mengubah permutasi yang digunakan berdasarkan query yang sudah diberikan program Anda.

Flushing. Jika Anda tidak menggunakan template yang disediakan, pastikan untuk melakukan **flush** pada standard output setelah mencetak setiap baris; jika tidak, program Anda bisa dinilai sebagai *Not correct*. Dalam Python, ini terjadi secara otomatis jika Anda menggunakan `input()` untuk membaca baris, dan Anda juga dapat menggunakan `print(..., flush=True)` untuk memaksa *flush*. Dalam C++, `cout << endl;` juga melakukan flush selain mencetak baris baru; jika menggunakan printf, gunakan `fflush(stdout)`.

Batasan

- $1 \leq T \leq 10$.
- N adalah 5 (untuk contoh), 8, 40, atau 2000.
- Untuk setiap kasus uji, Anda boleh menggunakan paling banyak 10 000 query.

Subsoal dan Penilaian

Program Anda akan diuji pada beberapa kasus yang dikelompokkan ke dalam subsoal. Untuk mendapatkan poin pada suatu subsoal, Anda harus menyelesaikan semua tes di dalamnya dengan benar.

- **Subsoal 0 [0 poin]**: Contoh ($N = 5$).
- **Subsoal 1 [9 poin]**: $N = 8$.
- **Subsoal 2 [11 poin]**: $N = 2000$, dan tamu 0 dan 1 selalu duduk bersebelahan.
- **Subsoal 3 [15 poin]**: $N = 40$.
- **Subsoal 4 [65 poin]**: $N = 2000$.

Untuk subsoal 1 dan 2, solusi yang berhasil menyelesaikan semua kasus uji dengan benar akan mendapatkan semua poin.

Untuk subsoal 3 dan 4, solusi Anda harus menyelesaikan semua kasus uji dengan benar untuk mendapatkan poin, dan poin Anda bergantung pada Q_s, jumlah query terbanyak yang dilakukan solusi Anda untuk menyelesaikan sebuah kasus uji. Misalkan $X_s = \max(1, Q_s/N)$. Poin untuk subsoal 3 dan 4 kemudian dihitung sebagai berikut:

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

Nilai poin_s dibulatkan ke bilangan bulat terdekat per subsoal, dan total poin Anda adalah jumlah dari nilai-nilai tersebut. Untuk mendapatkan poin penuh, Anda perlu menyelesaikan subsoal 3 hanya dengan 55 query dan subsoal 4 hanya dengan 2597 query. Contoh penilaian dan poin untuk subsoal 3 dan 4 ditunjukkan di bawah ini.

Q_s	55	56	60	70	80	100	150	10000
poin_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
poin_4	65	58	53	35	26	21	14	11

Contoh

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

Penjelasan

Contoh masukan berisi satu kasus uji ($T = 1$) dengan $N = 5$ tamu, dengan konfigurasi yang sesuai dengan Figur 1.

Query pertama yang dibuat oleh contoh solusi adalah 0, 2, 4. Jawaban 3 untuk query ini memberi tahu kita bahwa para tamu ini duduk di tiga kursi berturut-turut di sebelah satu sama lain.

Jawaban 3 untuk query kedua memberi tahu kita hal yang sama tentang tamu 3, 0, dan 1.

Kita dapat menyimpulkan bahwa tamu 0 harus duduk di tengah, dengan tamu 2 dan 4 berada di satu sisi dan tamu 1 dan 3 berada di sisi lainnya.

Setelah query ketiga, kita yakin bahwa para tamu harus duduk dalam urutan $[3, 1, 0, 2, 4]$ atau $[4, 2, 0, 1, 3]$ (urutan terbalik). Kita boleh mengeluarkan salah satu dari urutan tersebut.

Template dan Detail Evaluasi di CMS

Penggunaan kode *template* untuk C++ dan Python sangat direkomendasikan. Kode template tersebut bisa mengecek keberhasilan komunikasi dengan grader (dan bantu memberhentikan apabila tidak).

Jika Anda tidak menggunakan kode template yang diberikan dan apabila solusi Anda salah, CMS mungkin memberikan pesan yang salah. Sebagai contoh, CMS mungkin menampilkan “Execution killed by signal” atau “Execution timed out (wall clock limit exceeded)” dibandingkan “Output isn’t correct”.

Penggunaan alat pengujian (di bawah) untuk menguji solusi Anda secara lokal juga sangat direkomendasikan. Alat tersebut dapat mengecek keluaran dari solusi Anda dan melaporkan penggunaan query yang tidak valid.

Alat Pengujian

Untuk memfasilitasi pengujian solusi Anda, terdapat alat yang bisa Anda unduh dari CMS. Alat ini bersifat opsional. Perhatikan bahwa grader resmi di CMS berbeda dengan alat pengujian.

Untuk menggunakan alat tersebut, Anda memerlukan berkas masukan. Anda bisa menggunakan contoh masukan yang disediakan `seatingplan.input0.txt` atau membuat milik Anda sendiri. Berkas masukan harus dimulai dengan baris yang berisi jumlah kasus uji T dan kemudian harus memiliki dua baris per kasus uji: satu baris dengan angka N kemudian satu baris lainnya dengan urutan $g_0, g_1, ..., g_{N-1}$.

Untuk program Python, katakanlah `seatingplan.py` (biasanya dijalankan sebagai `pypy3 seatingplan.py`) jalankan alat pengujian sebagai berikut:

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

Untuk program C++, pertama-tama kompilasi solusi Anda:

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

dan kemudian jalankan alat pengujian:

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