D. Plan de Asientos (seatingplan)

La ceremonia de clausura de esta EGOI contará con la presencia de N invitadas importantes. Todas ellas deben sentarse en la primera fila en un orden muy específico para respetar las complejas reglas burocráticas del protocolo diplomático. Determinar el orden correcto de los asientos le tomó a Noemi dos noches sin dormir.

Verónica está supervisando la ceremonia de clausura. Una de sus muchas responsabilidades es asegurarse de que los asientos de la primera fila tengan las tarjetas con los nombres correctos. Solo hay un pequeño problema: Noemi nunca le dijo el orden correcto de los asientos, y ahora Noemi no aparece por ningún lado. Por suerte, Dorka la fotógrafa tiene una aplicación que podría ser útil.

Dorka tuvo que preparar sus cámaras para poder tomar algunas fotos específicas de las invitadas en la primera fila. Para la configuración, necesitaba saber qué tan ancha sería cada foto, así que Noemi le hizo una aplicación que arroja rápidamente la información que necesita. Verónica ahora quiere usar la aplicación para saber como asignar los asientos correctamente.

Las N invitadas importantes están numerados del 0 al $N-1$. Los asientos de la primera fila también están numerados del 0 al $N-1$, de izquierda a derecha. Para cada I $(0 \leq I \leq N-1)$, sea g_I la invitada que debe sentarse en el asiento I, y sea s_I el asiento en el que debe sentarse la invitada I.

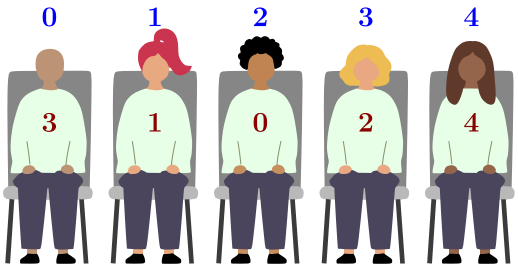

Figura 1: Una fila con cinco invitadas. Para esta fila, $g = [3, 1, 0, 2, 4]$ y $s = [2, 1, 3, 0, 4]$.

La aplicación funciona de la siguiente manera:

- Dorka ingresa los números I, J, K de exactamente tres invitadas diferentes.
- La aplicación le indica el número mínimo de invitadas que saldran en la foto si se asegura que las tres invitadas seleccionados estarán en la foto. Formalmente, la aplicación mostrará el valor $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Por ejemplo, mira la situación mostrada en la Figura 1:

- Las invitadas $I = 0$, $J = 2$ y $K = 4$ están en los asientos $s_I = 2$, $s_J = 3$ y $s_K = 4$. Si Dorka los selecciona, la aplicación mostrará el valor $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  En otras palabras, la foto más estrecha que contiene a las invitadas 0, 2 y 4 tiene solo a estas tres invitadas.

- Las invitadas $I = 0$, $J = 4$ y $K = 3$ están en los asientos $s_I = 2$, $s_J = 4$ y $s_K = 0$. Si Dorka los selecciona, la aplicación mostrará el valor $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  En otras palabras, una foto que contiene a las tres invitadas seleccionadas debe contener a las 5 invitadas.

Ayuda a Verónica a determinar el orden correcto de los asientos usando la aplicación de Dorka. Más específicamente, tu programa debe determinar y mostrar la secuencia $g_0, g_1, ..., g_{N-1}$. Siempre hay exactamente dos respuestas correctas (una siendo la inversa de la otra), y puedes mostrar cualquiera de ellas. Tu puntaje dependerá del número de consultas a la aplicación que haga tu solución.

Implementación

⇨ Este es un problema interactivo. Tu programa usará la entrada y salida estándar para comunicarse con un evaluador en el formato descrito a continuación.

Tu programa debe comenzar leyendo una línea de entrada que contenga un entero positivo T, el número de casos de prueba que siguen.

Para cada caso de prueba, tu programa debe comenzar leyendo una línea de entrada que contenga un entero positivo N, el número de asientos, que también es el número de invitadas.

Para hacer una consulta, tu programa debe mostrar una línea de la forma "? I J K", donde $0 \leq I, J, K \leq N-1$ son tres números **distintos**.

Después de hacer una consulta, tu programa debe leer una línea que contenga un entero positivo, la respuesta a tu consulta.

Para responder con un orden de asientos correcto, tu programa debe mostrar una línea de la forma "! g_0 ⋯ g_{N-1}".

Después de resolver todos los casos de prueba T, tu programa debe terminar normalmente.

Toma en cuenta que el evaluador oficial utilizado en CMS para probar tu solución puede ser **adaptativo**. Es decir, para algunos casos de prueba, la permutación de las invitadas no se determina de antemano. En cambio, el evaluador puede decidir cuál de las permutaciones restantes usar dependiendo de las consultas que tu programa ya haya hecho.

Limpieza de búfer (Flushing). Si no estás utilizando las plantillas proporcionadas, asegúrate de limpiar la salida estándar después de imprimir cada línea; de lo contrario, tu programa podría trabarse y ser calificado como *No correcto*. En Python, esto sucede automáticamente si usas `input()` para leer líneas, y puedes usar `print(..., flush=True)` para forzar el vaciado. En C++, `cout << endl;` limpia el búfer además de imprimir un salto de línea; si usas printf, usa `fflush(stdout)`.

Restricciones

- $1 \leq T \leq 10$.
- N será 5 (solo ejemplo), 8, 40 o 2000.
- Para cada caso de prueba, puedes hacer como máximo 10 000 consultas.

Puntuación

Tu programa será probado en varios casos de prueba agrupados en subtareas. Para obtener el puntaje de una subtarea, debes resolver correctamente todas las pruebas que contiene.

- **Subtarea 0 [0 puntos]**: Ejemplo ($N = 5$).
- **Subtarea 1 [9 puntos]**: $N = 8$.
- **Subtarea 2 [11 puntos]**: $N = 2000$, y las invitadas 0 y 1 se sientan una al lado de la otra.
- **Subtarea 3 [15 puntos]**: $N = 40$.
- **Subtarea 4 [65 puntos]**: $N = 2000$.

Para las subtareas 1 y 2, cualquier solución que resuelva correctamente todos los casos de prueba obtendrá todos los puntos.

Para las subtareas 3 y 4, tu solución debe resolver correctamente todos los casos de prueba para obtener puntos, y tu puntaje dependerá de Q_s, el número máximo de consultas que tu solución necesitó hacer para resolver un caso de prueba. Sea $X_s = \max(1, Q_s/N)$. Los puntajes para las subtareas 3 y 4 se calculan de la siguiente manera:

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

El valor de puntaje_s se redondea al número entero más cercano por subtarea, y tu puntaje total es la suma de estos. Para obtener el puntaje completo, necesitas resolver la subtarea 3 en un máximo de 55 consultas y la subtarea 4 en un máximo de 2597 consultas. A continuación se muestran valores de ejemplo de Q_s y los puntajes para las subtareas 3 y 4.

Q_s	55	56	60	70	80	100	150	10000
puntaje_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
puntaje_4	65	58	53	35	26	21	14	11

Ejemplos

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

Explicación

La entrada de ejemplo contiene un caso de prueba ($T = 1$) con $N = 5$ invitadas. La configuración oculta de las invitadas en este caso de prueba corresponde a Figura 1.

La primera consulta realizada por la solución de ejemplo es 0, 2, 4. La respuesta 3 a esta consulta nos dice que estas invitadas se sientan, en algún orden desconocido, en tres asientos consecutivos una al lado de la otra.

La respuesta 3 a la segunda consulta nos dice lo mismo sobre las invitadas 3, 0 y 1.

Ahora podemos deducir que la invitada 0 debe sentarse en el medio, con las invitadas 2 y 4 de un lado y las invitadas 1 y 3 del otro lado.

Después de la tercera consulta, ya podemos estar seguros de que las invitadas deben sentarse ya sea en el orden $[3, 1, 0, 2, 4]$ o en el orden inverso $[4, 2, 0, 1, 3]$. Podemos mostrar cualquiera de los dos órdenes.

Plantillas de código y detalles de evaluación en CMS

Recomendamos ampliamente usar las plantillas de código proporcionadas para C++ y Python. Estas verifican si la comunicación con el evaluador fue exitosa y terminan correctamente cuando no lo fue.

Si no usas las plantillas proporcionadas, en casos donde tu solución sea incorrecta, es posible que CMS muestre un veredicto incorrecto. Por ejemplo, en lugar de «Output isn't correct» podrías recibir «Execution killed by signal» o «Execution timed out (wall clock limit exceeded)».

También recomendamos la herramienta de prueba (ver más abajo) para probar tu solución localmente antes de enviarla. La herramienta de prueba verifica las salidas de tu solución e informa sobre el uso de consultas no válidas.

Herramienta de prueba

Para facilitar la prueba de tu solución, proporcionamos una herramienta sencilla que puedes descargar desde CMS. El uso de la herramienta es opcional. Toma en cuenta que el evaluador oficial en CMS es diferente de la herramienta de prueba.

Para usar la herramienta, necesitas un archivo de entrada. Puedes usar la entrada de ejemplo proporcionada `seatingplan.input0.txt` o crear la tuya propia. El archivo de entrada debe comenzar con una línea que tenga el número T de casos de prueba y luego debe tener dos líneas por caso de prueba: una línea con el número N y luego una línea con los números $g_0, g_1, \ldots, g_{N-1}$.

Para programas en Python, digamos `seatingplan.py` (que normalmente se ejecuta como `pypy3 seatingplan.py`) ejecuta la herramienta de prueba de la siguiente manera:

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

Para programas en C++, primero compila tu solución:

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

y luego ejecuta la herramienta de prueba:

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