D. Plan de Asientos (seatingplan)

La ceremonia de clausura de esta EGOI contará con la presencia de N invitados importantes. Todos ellos deben sentarse en la primera fila en un orden muy específico que corresponde a todos los matices del protocolo diplomático. Determinar el orden correcto de los asientos le tomó a Noemi dos noches sin dormir.

Veronica está supervisando la ceremonia de clausura. Una de sus muchas responsabilidades es asegurarse de que los asientos de la primera fila tengan los letreros con nombres correctos. Solo hay un pequeño problema: Noemi nunca le dijo cuál era el orden correcto de los asientos, y ahora Noemi no aparece por ninguna parte. 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 los invitados de la primera fila. Para la preparación, necesitaba saber qué tan ancha sería cada foto, así que Noemi le hizo una aplicación que rápidamente muestra la información que necesita. Ahora Veronica quiere usar la aplicación para encontrar la asignación correcta de los asientos.

Los N invitados 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 el invitado que debe sentarse en el asiento I, y sea s_I el asiento en el que debe sentarse el invitado I.

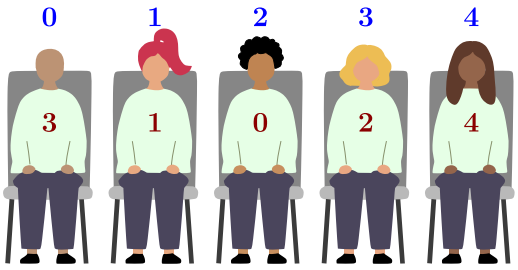

Figura 1: Una fila con cinco invitados. Para esta fila, $g = [3, 1, 0, 2, 4]$ y $s = [2, 1, 3, 0, 4]$.

La aplicación funciona de la siguiente manera:

- Dorka introduce los números I, J, K de exactamente tres invitados diferentes.
- La aplicación le indica el número mínimo de invitados que serán visibles si los tres invitados seleccionados aparecen 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, observa la situación mostrada en Figura 1:

- Los invitados $I = 0$, $J = 2$ 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 los invitados 0, 2 y 4 contiene únicamente a esos tres invitados.

- Los invitados $I = 0$, $J = 4$ 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 contenga a los tres invitados dados debe contener a los 5 invitados.

Ayuda a Veronica 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 existen exactamente dos respuestas correctas (una es la inversa de la otra), y puedes mostrar cualquiera de ellas. Tu puntuación dependerá del número de consultas a la aplicación que realice tu solución.

Implementación

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

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

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

Para realizar una consulta, tu programa debe imprimir una línea con el formato «? I J K», donde $0 \leq I, J, K \leq N-1$ son tres números **distintos**.

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

Para responder con un orden correcto de asientos, tu programa debe imprimir una línea con el formato «! g_0 … g_{N-1}».

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

Ten en cuenta que el juez oficial utilizado en CMS para evaluar tu solución puede ser **adaptativo**. Es decir, para algunos casos de prueba, la permutación de invitados no está determinada de antemano. En su lugar, el juez puede decidir cuál de las permutaciones restantes utilizar dependiendo de las consultas ya realizadas por tu programa.

Vaciado del búfer (Flushing). Si no estás utilizando las plantillas proporcionadas, asegúrate de vaciar la salida estándar después de imprimir cada línea; de lo contrario, tu programa podría recibir el veredicto *Not correct*. En Python, esto ocurre automáticamente si utilizas `input()` para leer líneas, y puedes usar `print(..., flush=True)` para forzar el vaciado. En C++, `cout << endl;` vacía el búfer además de imprimir un salto de línea; si usas `printf`, utiliza `fflush(stdout)`.

Restricciones

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

Puntuación

Tu programa será evaluado en varios casos de prueba agrupados en subtareas. Para obtener la puntuación de una subtarea, debes resolver correctamente todos los casos que contiene.

- **Subtask 0 [0 puntos]**: Ejemplo ($N = 5$).
- **Subtask 1 [9 puntos]**: $N = 8$.
- **Subtask 2 [11 puntos]**: $N = 2000$, y los invitados 0 y 1 están sentados uno al lado del otro.
- **Subtask 3 [15 puntos]**: $N = 40$.
- **Subtask 4 [65 puntos]**: $N = 2000$.

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

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

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

El valor de score_s se redondea al entero más cercano por subtarea, y tu puntuación total es la suma de estas. Para obtener la puntuación máxima, necesitas resolver la subtarea 3 en como máximo 55 consultas y la subtarea 4 en como máximo 2597 consultas. A continuación se muestran ejemplos de valores de Q_s y puntuaciones para las subtareas 3 y 4.

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

Ejemplos de entrada/salida

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 $N = 5$ invitados. La configuración oculta de invitados 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 estos invitados están sentados, en algún orden desconocido, en tres asientos consecutivos.

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

Ahora podemos deducir que el invitado 0 debe sentarse en el medio, con los invitados 2 y 4 a un lado y los invitados 1 y 3 al otro lado.

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

Plantillas de Código y Detalles de Evaluación en CMS

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

Si no utilizas las plantillas proporcionadas, en los casos donde tu solución sea incorrecta, CMS podría mostrar 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 usar la herramienta de prueba (ver abajo) para probar tu solución localmente antes de enviarla. La herramienta de prueba verifica las salidas de tu solución y reporta el uso de consultas inválidas.

Herramienta de Prueba

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

Para usar la herramienta, necesitas un archivo de entrada. Puedes usar el archivo de ejemplo proporcionado `seatingplan.input0.txt` o crear uno propio. El archivo de entrada debe comenzar con una línea que contenga 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 $g_0, g_1, ..., g_{N-1}$.

Para programas en Python, por ejemplo `seatingplan.py` (normalmente ejecutado 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