

## D. Ospiti Importanti (seatingplan)

Alla cerimonia di chiusura di queste EGOI parteciperanno  $N$  ospiti importanti. Tutti devono sedersi in prima fila in un ordine molto preciso che rispetta tutte le sottigliezze del protocollo diplomatico. Determinare il giusto ordine ha portato Noemi a due notti insonni.

Veronica sta supervisionando la cerimonia di chiusura. Una delle sue tante responsabilità è assicurarsi che i cartellini con i nomi sui posti della prima fila siano corretti. C'è solo un piccolo problema: Noemi non le ha mai detto l'ordine corretto dei posti e ora Noemi è introvabile. Per fortuna, la fotografa Daria ha un'app che potrebbe tornare utile.

Daria doveva preparare le sue fotocamere per poter scattare alcune foto specifiche agli ospiti in prima fila. Per la configurazione, doveva sapere quanto sarebbe stata larga ogni foto, quindi Noemi ha creato per lei un'app che restituisce rapidamente le informazioni di cui ha bisogno. Veronica ora vuole usare l'app per trovare l'assegnazione corretta dei posti.

Gli  $N$  ospiti importanti sono numerati da 0 a  $N - 1$ . I posti nella prima fila sono anch'essi numerati da 0 a  $N - 1$ , da sinistra a destra. Per ogni  $I$  ( $0 \leq I \leq N - 1$ ), sia  $g_I$  l'ospite che dovrebbe sedersi al posto  $I$ , e sia  $s_I$  il posto in cui l'ospite  $I$  dovrebbe sedersi.

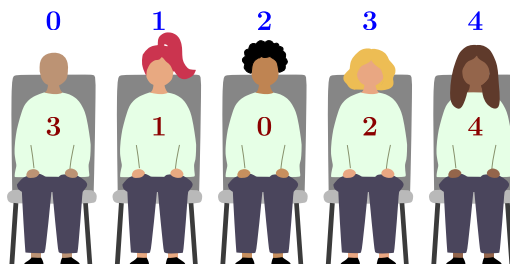


Figura 1: Una fila con cinque ospiti. Per questa fila,  $g = [3, 1, 0, 2, 4]$  e  $s = [2, 1, 3, 0, 4]$ .

L'app funziona come segue:

- Daria inserisce i numeri  $I$ ,  $J$ ,  $K$  di esattamente tre ospiti diversi.
- L'app le dice il minimo numero di ospiti che saranno visibili se tutti e tre gli ospiti selezionati sono nella foto. Formalmente, l'app mostrerà il valore  $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$ .

Per esempio, facendo riferimento alla situazione mostrata in Figura 1:

- Gli ospiti  $I = 0$ ,  $J = 2$  e  $K = 4$  sono nei posti  $s_I = 2$ ,  $s_J = 3$  e  $s_K = 4$ . Se Daria li seleziona, l'app mostrerà il valore  $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$ .

In altre parole, la foto più stretta che contiene gli ospiti 0, 2 e 4 include solo questi tre ospiti.

- Gli ospiti  $I = 0$ ,  $J = 4$  e  $K = 3$  sono nei posti  $s_I = 2$ ,  $s_J = 4$  e  $s_K = 0$ . Se Daria li seleziona, l'app mostrerà il valore  $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$ .

In altre parole, una foto che contiene i tre ospiti dati deve contenere tutti e 5 gli ospiti.

Aiuta Veronica a determinare l'ordine corretto dei posti usando l'app di Daria. Più specificamente, il tuo programma dovrebbe determinare e stampare la sequenza  $g_0, g_1, \dots, g_{N-1}$ . Ci sono sempre esattamente due risposte corrette (una è il riflesso dell'altra), e puoi stampare una qualsiasi di esse. Il tuo punteggio dipenderà dal numero di query all'app fatte dalla tua soluzione.

## Implementazione



Questo è un problema interattivo. Il tuo programma userà lo standard input e output per comunicare con un grader nel formato descritto in seguito.

Il tuo programma deve leggere una riga di input contenente un intero positivo  $T$ , il numero di casi di test che seguono.

Per ogni caso di test, il tuo programma deve leggere una riga di input contenente un intero positivo  $N$ , il numero di posti, che è anche il numero di ospiti.

Per fare una query, il tuo programma deve stampare una riga della forma «?  $I$   $J$   $K$ », dove  $0 \leq I, J, K \leq N - 1$  sono tre numeri **distinti**.

Dopo aver fatto una query, il tuo programma deve leggere una riga contenente un intero positivo, la risposta alla tua query.

Per rispondere con l'ordine corretto dei posti, il tuo programma deve stampare una riga della forma «!  $g_0 \dots g_{N-1}$ ».

Dopo aver risolto tutti i  $T$  casi di test, il tuo programma deve terminare normalmente.

Nota che il grader ufficiale usato su CMS per testare la tua soluzione potrebbe essere **adattivo**. Ovvero, per alcuni casi di test, la permutazione degli ospiti non è determinata in anticipo. Invece, il grader potrebbe decidere quale delle rimanenti permutazioni usare a seconda delle query già fatte dal tuo programma.

**Flushing.** Se non stai usando i template forniti, assicurati di effettuare il flush dello standard output dopo aver stampato ogni riga, altrimenti il tuo programma potrebbe essere giudicato come *Non corretto*. In Python, questo accade automaticamente se usi `input()` per leggere le righe, e puoi usare `print(..., flush=True)` per forzare il flush. In C++, `cout << endl;` effettua il flush oltre a stampare una newline; se usi `printf`, usa `fflush(stdout)`.

## Assunzioni

- $1 \leq T \leq 10$ .
- $N$  sarà 5 (solo negli esempi), 8, 40 o 2000.
- Per ogni caso di test, puoi fare al massimo 10 000 query.

## Assegnazione del punteggio

Il tuo programma sarà testato su diversi casi di test raggruppati in subtask. Per ottenere il punteggio di un subtask, devi risolvere correttamente tutti i test che contiene.

- **Subtask 0** [ **0 punti**]: Casi d'esempio ( $N = 5$ ).
- **Subtask 1** [ **9 punti**]:  $N = 8$ .
- **Subtask 2** [ **11 punti**]:  $N = 2000$  e gli ospiti 0 e 1 siedono vicini.
- **Subtask 3** [ **15 punti**]:  $N = 40$ .
- **Subtask 4** [ **65 punti**]:  $N = 2000$ .

Per i subtask 1 e 2, qualsiasi soluzione che risolve correttamente tutti i casi di test riceverà tutti i punti.

Per i subtask 3 e 4, la tua soluzione deve risolvere correttamente tutti i casi di test per ottenere punti, e il tuo punteggio dipenderà da  $Q_s$ , il numero massimo di query che la tua soluzione ha dovuto fare per risolvere un caso di test. Sia  $X_s = \max(1, Q_s/N)$ . I punteggi per i subtask 3 e 4 sono quindi calcolati come segue:

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

Il valore di  $\text{score}_s$  è arrotondato all'intero più vicino per subtask, e il tuo punteggio totale è la somma di questi. Per ottenere il punteggio pieno, devi risolvere il subtask 3 in al massimo 55 query e il subtask 4 in al massimo 2597 query. Esempi di valori di  $Q_s$  e punteggi per i subtask 3 e 4 sono mostrati qui sotto.

$Q_s$	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

## Esempi di input/output

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

## Spiegazione

L'input di esempio contiene un caso di test ( $T = 1$ ) con  $N = 5$  ospiti. La configurazione nascosta degli ospiti in questo caso di test corrisponde a Figura 1.

La prima query fatta dalla soluzione di esempio è 0, 2, 4. La risposta 3 a questa query ci dice che questi ospiti siedono, in un ordine sconosciuto, in tre posti consecutivi vicini.

La risposta 3 alla seconda query ci dice lo stesso riguardo agli ospiti 3, 0 e 1.

Possiamo ora dedurre che l'ospite 0 deve sedersi nel mezzo, con gli ospiti 2 e 4 da un lato e gli ospiti 1 e 3 dall'altro.

Dopo la terza query, possiamo già essere sicuri che gli ospiti debbano sedersi nell'ordine [3, 1, 0, 2, 4] o nell'ordine riflesso [4, 2, 0, 1, 3]. Possiamo stampare uno qualsiasi dei due ordini.

## Template di Codice e Dettagli di Valutazione di CMS

Consigliamo fortemente di usare i template di codice forniti per C++ e Python. Questi controllano se la comunicazione con il grader è andata a buon fine e terminano correttamente quando non lo è.

Se non usi i template forniti, nei casi in cui la tua soluzione è errata, CMS potrebbe mostrare il verdetto sbagliato. Per esempio, invece di «Output isn't correct» potresti ricevere «Execution killed by signal» o «Execution timed out (wall clock limit exceeded)».

Consigliamo anche il tool di test (vedi sotto) per testare la tua soluzione localmente prima di inviarla. Il tool di test controlla gli output della tua soluzione e segnala l'uso di query non valide.

## Tool di Test

Per facilitare il test della tua soluzione, forniamo un semplice tool che puoi scaricare da CMS. L'uso del tool è facoltativo. Nota che il grader ufficiale su CMS è diverso dal tool di test.

Per usare il tool, hai bisogno di un file di input. Puoi usare l'input di esempio fornito `seatingplan.input0.txt` o crearne uno tuo. Il file di input dovrebbe iniziare con una riga che contiene il numero  $T$  di casi di test e poi dovrebbe avere due righe per caso di test: una riga con il numero  $N$  e poi una riga con i numeri  $g_0, g_1, \dots, g_{N-1}$ .

Per i programmi Python, diciamo `seatingplan.py` (normalmente eseguito come `pypy3 seatingplan.py`) esegui il tool di test come segue:

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

Per i programmi C++, compila prima la tua soluzione:

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

e poi esegui il tool di test:

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