D. Plano de Lugares (seatingplan)

A cerimónia de encerramento desta EGOI terá a presença de N convidados importantes. Todos têm de ficar sentados na primeira fila numa ordem muito específica que corresponde a todas as nuances do protocolo diplomático. Determinar a ordem correta dos lugares custou à Noemi duas noites sem dormir.

A Veronica está a supervisionar a cerimónia de encerramento. Uma das suas muitas responsabilidades é garantir que os lugares da primeira fila têm as etiquetas com o nome correto. Existe apenas um pequeno problema: a Noemi nunca lhe disse a ordem correta dos lugares e, agora, não dá para encontrar a Noemi em lado nenhum. Felizmente, a Dorka, a fotógrafa, tem uma app que pode ser útil.

A Dorka teve de preparar as suas câmaras para poder tirar algumas fotos específicas aos convidados na primeira fila. Para as configurar, ela precisava de saber qual seria a largura de cada foto, por isso a Noemi fez-lhe uma app que fornece rapidamente a informação de que ela precisa. A Veronica quer agora usar a app para encontrar a atribuição correta dos lugares.

Os N convidados importantes estão numerados de 0 a $N-1$. Os lugares na primeira fila também estão numerados de 0 a $N-1$, da esquerda para a direita. Para cada I $(0 \leq I \leq N-1)$, seja g_I o convidado que deve sentar-se no lugar I, e seja s_I o lugar onde o convidado I deve sentar-se.

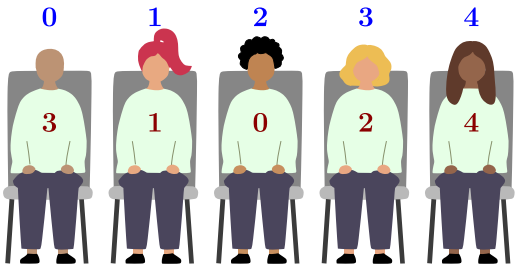

Figura 1: Uma fila com cinco convidados. Para esta fila, $g = [3, 1, 0, 2, 4]$ e $s = [2, 1, 3, 0, 4]$.

A app funciona da seguinte forma:

- A Dorka introduz exatamente três números I, J, K de três convidados diferentes.
- A app diz-lhe o número mínimo de convidados que ficarão visíveis se todos os três convidados selecionados estiverem na foto. Formalmente, a app irá mostrar o valor $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Por exemplo, vê a situação mostrada em Figura 1:

- Os convidados $I = 0$, $J = 2$ e $K = 4$ estão nos lugares $s_I = 2$, $s_J = 3$ e $s_K = 4$. Se a Dorka os selecionar, a app mostrará o valor $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Por outras palavras, a foto mais estreita que contém os convidados 0, 2 e 4 tem apenas estes três convidados.

- Os convidados $I = 0$, $J = 4$ e $K = 3$ estão nos lugares $s_I = 2$, $s_J = 4$ e $s_K = 0$. Se a Dorka os selecionar, a app mostrará o valor $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Por outras palavras, uma foto que contenha os três convidados dados tem de conter todos os 5 convidados.

Ajuda a Veronica a determinar a ordem correta dos lugares usando a app da Dorka. Mais especificamente, o teu programa deve determinar e imprimir a sequência $g_0, g_1, ..., g_{N-1}$. Existem sempre exatamente duas respostas corretas (uma sendo o inverso da outra), e podes imprimir qualquer uma delas. A tua pontuação dependerá do número de consultas que a tua solução faz à app.

Implementação

⇨ Este é um problema interativo. O teu programa utilizará o standard input e standard output para comunicar com um avaliador no formato descrito abaixo.

O teu programa deve começar por ler uma linha de input contendo um inteiro positivo T, o número de casos de teste que se seguem.

Para cada caso de teste, o teu programa deve começar por ler uma linha de input contendo um inteiro positivo N, o número de lugares, que é também o número de convidados.

Para fazer uma consulta, o teu programa deve imprimir uma linha com o formato "? I J K", onde $0 \leq I, J, K \leq N-1$ são três números **distintos**.

Após fazer uma consulta, o teu programa deve ler uma linha contendo um inteiro positivo, a resposta à tua consulta.

Para responder com a ordem correta dos lugares, o teu programa deve imprimir uma linha com o formato "! g_0 … g_{N-1}".

Após resolver todos os T casos de teste, o teu programa deve terminar normalmente.

Nota que o avaliador oficial usado no CMS para testar a tua solução pode ser **adaptativo**. Isto é, para alguns casos de teste, a permutação dos convidados não é determinada com antecedência. Em vez disso, o avaliador pode decidir qual das permutações restantes ele usa dependendo das consultas já feitas pelo teu programa.

Limpeza de buffer (Flushing). Se não estiveres a usar os templates fornecidos, certifica-te de que limpas o buffer (flush) do standard output após imprimir cada linha, caso contrário o teu programa poderá ser avaliado como *Not correct*. Em Python, isto acontece automaticamente se usares `input()` para ler linhas, e podes usar `print(..., flush=True)` para forçar a dar flush. Em C++, `cout << endl;` limpa o buffer além de imprimir uma mudança de linha; se usares printf, usa `fflush(stdout)`.

Restrições

- $1 \leq T \leq 10$.
- N será 5 (apenas exemplo), 8, 40, ou 2000.
- Para cada caso de teste, podes fazer no máximo 10 000 consultas.

Pontuação

O teu programa será testado em vários casos de teste agrupados em subtarefas. Para obter a pontuação de uma subtarefa, tens de resolver corretamente todos os testes que ela contém.

- **Subtarefa 0 [0 pontos]**: Exemplo ($N = 5$).
- **Subtarefa 1 [9 pontos]**: $N = 8$.
- **Subtarefa 2 [11 pontos]**: $N = 2000$, e os convidados 0 e 1 sentam-se um ao lado do outro.
- **Subtarefa 3 [15 pontos]**: $N = 40$.
- **Subtarefa 4 [65 pontos]**: $N = 2000$.

Para as subtarefas 1 e 2, qualquer solução que resolva corretamente todos os casos de teste receberá a pontuação total.

Para as subtarefas 3 e 4, a tua solução deve resolver corretamente todos os casos de teste para pontuar, e a tua pontuação dependerá de Q_s, o maior número de consultas que a tua solução precisou de fazer para resolver um caso de teste. Seja $X_s = \max(1, Q_s/N)$. As pontuações para as subtarefas 3 e 4 são então calculadas da seguinte forma:

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

O valor de score_s é arredondado ao número inteiro mais próximo por subtarefa, e a tua pontuação total é a soma destas. Para obteres a pontuação total, precisas de resolver a subtarefa 3 em, no máximo, 55 consultas e a subtarefa 4 em, no máximo, 2597 consultas. Exemplos de valores de Q_s e pontuações para as subtarefas 3 e 4 são mostrados abaixo.

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

Exemplos

Avaliador	Solução
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Explicação

O input do exemplo contém um caso de teste ($T = 1$) com $N = 5$ convidados. A configuração oculta dos convidados neste caso de teste corresponde à Figura 1.

A primeira consulta feita pela solução de exemplo é 0, 2, 4. A resposta 3 a esta consulta diz-nos que estes convidados se sentam, numa ordem desconhecida, em três lugares consecutivos adjacentes.

A resposta 3 à segunda consulta diz-nos o mesmo sobre os convidados 3, 0 e 1.

Podemos agora deduzir que o convidado 0 deve sentar-se no meio, com os convidados 2 e 4 de um lado e os convidados 1 e 3 do outro lado.

Após a terceira consulta, podemos ter a certeza de que os convidados devem sentar-se na ordem $[3, 1, 0, 2, 4]$ ou na ordem inversa $[4, 2, 0, 1, 3]$. Podemos imprimir qualquer uma das ordens.

Templates de Código e Detalhes de Avaliação no CMS

Recomendamos fortemente a usar os templates de código para C++ e Python. Estes verificam se o avaliador teve sucesso e acabam graciosamente se tal não acontecer.

Se não usares os templates providenciados, nos casos em que a tua solução não é correta, o CMS pode mostrar o verdito errado. Por exemplo, em vez de "Output isn't correct" poderás receber "Execution killed by signal" ou "Execution timed out (wall clock limit exceeded)".

Se a tua solução violar o protocolo (por exemplo faz uma consulta inválida ou supera o número de consultas permitidas), o avaliador que comunica com a tua solução vai reportar o erro e terminar, mas a tua solução poderá não o fazer. O avaliador a terminar poderá fazer o teu programa crashar ou ficar à espera duma resposta e por isso, em vez da esperada resposta "Output isn't correct" vais provavelmente receber outros erros como "Execution timed out (wall clock limit exceeded)" porque o crash ou o limite de tempo superado foram reportados mais cedo. Por favor tem isso em conta quando interpretas as mensagens de erro no CMS.

Aconselhamos também a usares a ferramenta de teste providenciada (ver abaixo) para correres a tua solução localmente antes de a submeteres. A ferramenta de teste verifica o output da tua solução e reporta violações do protocolo.

Ferramenta de Teste

Para facilitar o teste da tua solução, fornecemos uma ferramenta simples que podes descarregar do CMS. A utilização da ferramenta é opcional. Nota que o avaliador oficial no CMS é diferente da ferramenta de teste.

Para usar a ferramenta, precisas de um ficheiro de input. Podes usar o exemplo de input fornecido `seatingplan.input0.txt` ou criar o teu próprio. O ficheiro de input deve começar com uma linha que tenha o número T de casos de teste e depois deve ter duas linhas por caso de teste: uma linha com o número N e depois uma linha com os números $g_0, g_1, ..., g_{N-1}$.

Para programas em Python, digamos `seatingplan.py` (normalmente executado como `pypy3 seatingplan.py`) executa a ferramenta de teste da seguinte forma:

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

Para programas em C++, primeiro compila a tua solução:

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

e depois executa a ferramenta de teste:

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