D. Plano de Assentos (seatingplan)

A cerimônia de encerramento desta EGOI contará com a presença de N convidados importantes. Todos eles precisam ser acomodados na primeira fileira em uma ordem bem específica que corresponde a todas as nuances do protocolo diplomático. Determinar a ordem correta dos assentos custou à Noemi duas noites sem dormir.

Veronica está supervisionando a cerimônia de encerramento. Uma de suas muitas responsabilidades é garantir que os assentos da primeira fileira tenham as etiquetas de nome corretas. Há apenas um pequeno problema: Noemi nunca lhe contou a ordem correta dos assentos e, agora, Noemi não está em lugar nenhum. Felizmente, Dorka, a fotógrafa, tem um aplicativo que pode ser útil.

Dorka teve que preparar suas câmeras para poder tirar algumas fotos específicas dos convidados na primeira fileira. Para a configuração, ela precisava saber qual seria a largura de cada foto, então Noemi fez para ela um aplicativo que fornece rapidamente as informações de que ela precisa. Veronica agora quer usar o aplicativo para encontrar a distribuição correta dos assentos.

Os N convidados importantes são numerados de 0 a $N-1$. Os assentos na primeira fileira também são 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 no assento I, e seja s_I o assento em que o convidado I deve sentar.

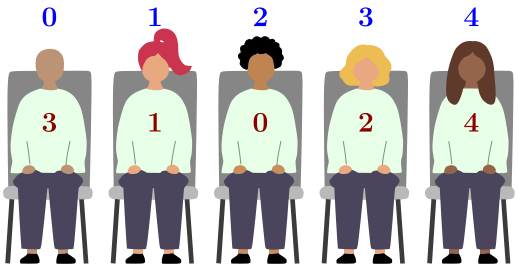

Figura 1: Uma fileira com cinco convidados. Para esta fileira, $g = [3, 1, 0, 2, 4]$ e $s = [2, 1, 3, 0, 4]$.

O aplicativo funciona da seguinte forma:

- Dorka entra com os números I, J, K de exatamente três convidados diferentes.
- O aplicativo informa o número mínimo de convidados que ficarão visíveis se todos os três convidados selecionados estiverem na foto. Formalmente, o aplicativo exibirá o valor $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Por exemplo, veja a situação mostrada em Figura 1:

- Os convidados $I = 0$, $J = 2$ e $K = 4$ estão nos assentos $s_I = 2$, $s_J = 3$ e $s_K = 4$. Se Dorka selecioná-los, o aplicativo exibirá o valor $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

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

- Os convidados $I = 0$, $J = 4$ e $K = 3$ estão nos assentos $s_I = 2$, $s_J = 4$ e $s_K = 0$. Se Dorka selecioná-los, o aplicativo exibirá o valor $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Em outras palavras, uma foto que contém os três convidados dados deve conter todos os 5 convidados.

Ajude a Veronica a determinar a ordem correta dos assentos usando o aplicativo da Dorka. Mais especificamente, seu programa deve determinar e imprimir a sequência $g_0, g_1, ..., g_{N-1}$. Sempre existem exatamente duas respostas corretas (uma sendo o inverso da outra), e você pode imprimir qualquer uma delas. Sua pontuação dependerá do número de consultas que sua solução fizer ao aplicativo.

Implementação

⇨ Este é um problema interativo. Seu programa usará a entrada e saída padrão para se comunicar com um corretor (grader) no formato descrito abaixo.

Seu programa deve começar lendo uma linha de entrada contendo um número inteiro positivo T, o número de casos de teste que se seguem.

Para cada caso de teste, seu programa deve começar lendo uma linha de entrada contendo um número inteiro positivo N, o número de assentos, que também é o número de convidados.

Para fazer uma consulta, seu programa deve imprimir uma linha no formato "? I J K", onde $0 \le I, J, K \le N-1$ são três números **distintos**.

Após fazer uma consulta, seu programa deve ler uma linha contendo um número inteiro positivo, a resposta para sua consulta.

Para responder com a ordem correta dos assentos, seu programa deve imprimir uma linha no formato "! g_0 ⋯ g_{N-1}".

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

Observe que o corretor oficial usado no CMS para testar sua solução pode ser **adaptativo**. Ou seja, para alguns casos de teste, a permutação dos convidados não é determinada de antemão. Em vez disso, o corretor pode decidir qual das permutações restantes ele usa dependendo das consultas já feitas pelo seu programa.

Flush (Limpeza de buffer). Se você não estiver usando os gabaritos (templates) fornecidos, certifique-se de realizar o flush da saída padrão após imprimir cada linha, caso contrário, seu programa poderá ser julgado como *Incorreto*. Em Python, isso acontece automaticamente se você usar `input()` para ler linhas, e você pode usar `print(..., flush=True)` para forçar o flush. Em C++, `cout << endl;` realiza o flush além de imprimir uma nova linha; se estiver usando printf, use `fflush(stdout)`.

Restrições

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

Pontuação

Seu programa será testado em vários casos de teste agrupados em subtarefas. Para obter a pontuação de uma subtarefa, você deve 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 lado a lado.
- **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 todos os pontos.

Para as subtarefas 3 e 4, sua solução deve resolver corretamente todos os casos de teste para pontuar, e sua pontuação dependerá de Q_s, o maior número de consultas que sua solução precisou 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 para o inteiro mais próximo por subtarefa, e sua pontuação total é a soma dessas. Para obter a pontuação máxima, você precisa resolver a subtarefa 3 em no máximo 55 consultas e a subtarefa 4 em no máximo 2597 consultas. Valores de exemplo 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 exemplo de entrada contém um caso de teste ($T = 1$) com $N = 5$ convidados. A configuração oculta dos convidados neste caso de teste corresponde a Figura 1.

A primeira consulta feita pela solução de exemplo é 0, 2, 4. A resposta 3 para esta consulta nos diz que esses convidados sentam, em alguma ordem desconhecida, em três assentos consecutivos, lado a lado.

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

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

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

Gabaritos (Templates) de Código e Detalhes de Avaliação no CMS

Recomendamos fortemente o uso dos gabaritos de código fornecidos para C++ e Python. Eles verificam se a comunicação com o corretor foi bem-sucedida e terminam de forma graciosa quando não foi.

Se você não usar os gabaritos fornecidos, em casos onde sua solução estiver incorreta, o CMS pode exibir um veredito errado. Por exemplo, em vez de "Output isn't correct" (Saída não está correta), você pode receber "Execution killed by signal" (Execução interrompida por sinal) ou "Execution timed out (wall clock limit exceeded)" (Tempo de execução esgotado).

Também recomendamos a ferramenta de teste (veja abaixo) para testar sua solução localmente antes de submetê-la. A ferramenta de teste verifica as saídas da sua solução e relata o uso de consultas inválidas.

Ferramenta de Teste

Para facilitar o teste da sua solução, fornecemos uma ferramenta simples que você pode baixar do CMS. O uso da ferramenta é opcional. Observe que o corretor oficial no CMS é diferente da ferramenta de teste.

Para usar a ferramenta, você precisa de um arquivo de entrada. Você pode usar o exemplo de entrada fornecido `seatingplan.input0.txt` ou criar o seu próprio. O arquivo de entrada deve começar com uma linha que contenha o número T de casos de teste e então deve ter duas linhas por caso de teste: uma linha com o número N e então 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`) execute a ferramenta de teste da seguinte forma:

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

Para programas em C++, primeiro compile sua solução:

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

e então execute a ferramenta de teste:

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