D. Plan de Salle (seatingplan)

N invitées importantes vont assister à la cérémonie de clôture de ces EGOI. Elles doivent toutes être assises au premier rang dans un ordre spécifique afin de correspondre à toutes les nuances du protocole diplomatique. Noémie n'a pas dormi des deux dernières nuits afin de déterminer le bon plan de salle.

Véronica supervise cette cérémonie. L'une de ses responsabilités est de vérifier que les noms inscrits sur les sièges du premier rang correspondent à celui de l'ordre. Il y a cependant un léger problème: Noémie n'a jamais donné l'ordre à Véronica et Noémie est introuvable. Fort heureusement, Dorka la photographe a une application qui pourrait être utile.

Dorka devait préparer ses caméras afin de prendre des photos spécifiques des invitées du premier rang. Pour cela, elle avait besoin de savoir pour chaque photo la largeur de la photo pour que toutes les personnes apparaissent dedans. Noémie lui a donc créé une application pour obtenir rapidement ces informations. Véronica veut maintenant essayer de retrouver le plan de salle à partir de cette application.

Il y a N invitées importantes numérotées de 0 à $N-1$. Les sièges du premier rang sont aussi numérotés de 0 à $N-1$. Pour chaque I $(0 \leq I \leq N-1)$, on définit g_I comme étant l'invitée qui est supposée s'asseoir sur le siège I, et on définit s_I comme étant le siège sur lequel l'invitée I est supposée s'asseoir.

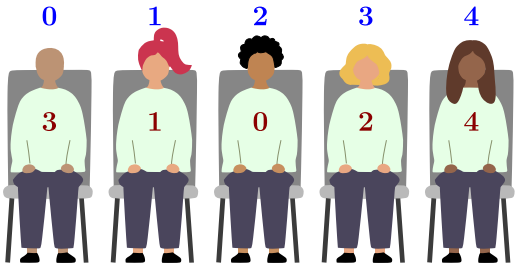

Fig. 1. – Une rangée avec cinq invitées. Pour cette rangée, $g = [3, 1, 0, 2, 4]$ et $s = [2, 1, 3, 0, 4]$.

L'application fonctionne de la façon suivante :

- Dorka entre trois nombres I, J, K de trois invitées distinctes.
- L'application lui indique alors le nombre minimum d'invitées qui seront visibles si les trois invitées sélectionnées sont sur la photo. Plus formellement, l'application affiche la valeur $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Par exemple, dans la situation de la Fig. 1:

- Les invitées $I = 0$, $J = 2$, et $K = 4$ sont sur les sièges $s_I = 2$, $s_J = 3$, et $s_K = 4$. Si Dorka les sélectionne, l'application affichera la valeur $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Dans d'autres termes, la plus petite photo contenant les invitées 0, 2 et 4 contient juste ces trois invitées.

- Les invitées $I = 0$, $J = 4$, et $K = 3$ sont dans les sièges $s_I = 2$, $s_J = 4$, and $s_K = 0$. Si Dorka les sélectionne, l'application affichera la valeur $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Dans d'autres termes, une photo qui contient ces trois invitées doit contenir toutes les 5 invitées.

Vous devez aider Véronica à déterminer le plan de salle en utilisant l'application de Dorka. Plus spécifiquement, votre programme doit déterminer et afficher la séquence $g_0, g_1, ..., g_{N-1}$.

Il y a toujours deux séquences correctes (l'une étant l'autre inversée), et vous pouvez afficher n'importe laquelle des deux. Votre score dépendera du nombre de requêtes que votre solution fait à l'application.

Implémentation

⇨ Ceci est un problème intéractif. Votre programme utilisera l'entrée standard et la sortie standard pour communiquer avec un grader dans le format décrit ci-dessous.

Votre programme doit commencer par lire une ligne de l'entrée contenant un entier positif T, le nombre de cas de tests qui suivront.

Pour chaque cas de test, votre programme doit commencer par lire une ligne de l'entrée contenant un entier positif N correspondant au nombre de sièges, qui est aussi le nombre d'invitées.

Pour faire une requête, votre programme doit afficher une ligne de la forme « ? I J K », où $0 \leq I, J, K \leq N-1$ sont trois nombres **distincts**.

Après avoir effectué une requête, votre programme doit lire une ligne de l'entrée contenant un entier positif, la réponse de votre requête.

Pour répondre avec l'ordre correct, votre programme doit afficher une ligne de la forme « ! g_0 … g_{N-1} ».

Après avoir résolu les T cas de tests, votre programme doit se terminer normalement.

Notez que le grader officiel utilisé dans CMS pour tester votre solution peut être **adaptive**. Cela signifie que, pour certains cas de tests, la permutation des invitées n'est pas déterminée à l'avance. Ainsi, le grader peut changer son comportement selon les requêtes déjà demandées par votre programme.

Flushing. Si vous n'utilisez pas la template donnée, n'oubliez pas de flush la sortie standard après avoir affiché chaque ligne, ou sinon votre programme pourrait être jugé comme *Not correct*. En Python, cela a lieu automatiquement si vous utilisez `input()` pour lire des lignes, et vous pouvez utiliser `print(..., flush=True)` pour forcer le flush. En C++, `cout << endl;` flush en plus d'afficher une nouvelle ligne. Si vous utilisez printf, utilisez `fflush(stdout)`.

Contraintes

- $1 \leq T \leq 10$.
- N sera 5 (pour les exemples seulement), 8, 40, ou 2000.
- Pour chaque cas de tests, vous pouvez faire au plus 10 000 requêtes.

Score

Votre programme sera évalué sur plusieurs tests regroupés en sous-tâches. Pour obtenir les points d'une sous-tâche, vous devez résoudre correctement tous les cas de tests qu'elle contient.

- **Sous-tâche 0 [0 points]**: Exemple ($N = 5$).
- **Sous-tâche 1 [9 points]**: $N = 8$.
- **Sous-tâche 2 [11 points]**: $N = 2000$, et les invitées 0 et 1 sont assises à côté l'une de l'autre.
- **Sous-tâche 3 [15 points]**: $N = 40$.
- **Sous-tâche 4 [65 points]**: $N = 2000$.

Pour les sous-tâches 1 et 2, toute solution qui résout correctement tous les cas de tests recevra tous les points.

Pour les sous-tâches 3 et 4, votre solution doit résoudre correctement tous les cas de tests pour recevoir des points, et votre score dépend de Q_s, le nombre maximum de requêtes que votre solution utilise pour résoudre un cas de tests. Soit $X_s = \max(1, Q_s/N)$. Les scores des sous-tâches 3 et 4 sont calculés de la façon suivante:

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

La valeur de score_s est arrondie à l'entier le plus proche par sous-tâche, et votre score total est la somme de ces scores. Pour obtenir tous les points d'une sous-tâche, vous devez résoudre la sous-tâche 3 en utilisant au plus 55 requêtes et la sous-tâche 4 en utilisant au plus 2597 requêtes. Des exemples de valeurs de Q_s et de scores pour les sous-tâches 3 et 4 sont montrés ci-dessus.

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

Exemples

Évaluateur	Solution
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Explication

L'exemple d'entrée contient un cas de test ($T = 1$) avec $N = 5$ invitées. La configuration cachée des invitées correspond à celle de la Fig. 1.

La première requête faite par la solution d'exemple est 0, 2, 4. La réponse 3 à cette requête nous indique que les 3 invitées sont assises dans un ordre inconnu dans trois sièges consécutifs.

La réponse 3 à la seconde requête nous indique la même chose pour les invitées 3, 0 et 1.

On peut donc en déduire que l'invitée 0 doit être assise au milieu, avec les invitées 2 et 4 d'un côté et les invitées 1 et 3 de l'autre.

Après la troisième requête, on peut déjà être sûr que les invitées doivent soit être assises dans l'ordre $[3, 1, 0, 2, 4]$ ou dans l'ordre inverse $[4, 2, 0, 1, 3]$. Vous pouvez afficher n'importe lequel de ces ordres.

Code Templates et détails d'évaluation dans CMS

Nous recommandons fortement d'utiliser les templates pour C++ et Python. Celles ci vérifient si la communication avec le grader était réussie et termine gracieusement si ce n'est pas le cas.

Si vous n'utilisez pas les templates fournies, dans le cas où votre solution n'est pas correcte, CMS pourrait afficher le mauvais verdict. Par exemple, au lieu de « Output isn't correct », vous pourriez recevoir « Execution killed by signal » ou « Execution timed out (wall clock limit exceeded) ».

Nous recommandons aussi d'utiliser l'outil de test fourni (voir ci-dessous) pour tester votre solution localement avant de la soumettre. L'outil de test vérifie les sorties de votre solution et rapporte l'utilisation de quêtes invalides.

Outil de test

Pour faciliter le test de votre solution, nous vous proposons un simple outil de test que vous pouvez télécharger sur CMS. Il n'est pas obligatoire d'utiliser l'outil. Notez que le grader officiel sur CMS est différent de l'outil de test.

Pour utiliser l'outil, vous avez besoin d'un fichier d'entrée. Vous pouvez utiliser l'exemple d'entre donné `seatingplan.input0.txt` ou faire votre propre fichier. Le fichier d'entrée doit commencer par une ligne avec un seul nombre T, le nombre de cas de tests. Ensuite, le fichier doit contenir deux lignes par cas de tests: une ligne avec le nombre N et ensuite une ligne avec les nombres $g_0, g_1, ..., g_{N-1}$.

Pour les programmes Python, par exemple `seatingplan.py` (normalement lancé avec `pypy3 seatingplan.py`), vous devez lancer l'outil de test avec:

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

Pour les programmes en C++, vous devez d'abord compiler votre solution:

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

Et ensuite vous pouvez lancer l'outil de test avec:

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