D. План рассадки (seatingplan)

На церемонии закрытия EGOI будут присутствовать N почетных гостей. Их необходимо рассадить в первом ряду в строго определенном порядке, соответствующем всем тонкостям дипломатического протокола. Ноэми потратила две бессонные ночи, чтобы определить правильный порядок рассадки.

Вероника курирует церемонию закрытия. Одна из ее многочисленных обязанностей — убедиться, что на сиденьях в первом ряду стоят правильные именные таблички. Есть лишь одна крошечная проблема: Ноэми так и не сообщила ей правильный порядок рассадки, и сейчас Ноэми нигде не найти. К счастью, у фотографа Дорки есть приложение, которое может пригодиться.

Дорке нужно подготовить камеры, чтобы сделать несколько специфических фотографий гостей в первом ряду. Для настройки ей нужно было знать ширину каждой фотографии, поэтому Ноэми сделала для нее приложение, которое быстро выводит нужную информацию. Теперь Вероника хочет использовать это приложение, чтобы найти правильное расположение мест.

N почетных гостей пронумерованы от 0 до $N-1$. Места в первом ряду также пронумерованы от 0 до $N-1$ слева направо. Для каждого I $(0 \le I \le N-1)$ обозначим g_I как гостя, который должен сидеть на месте I, а s_I — как место, на котором должен сидеть гость I.

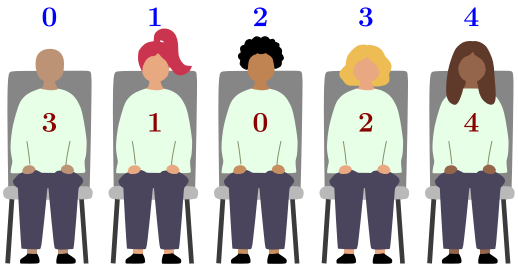

Рис. 1. Ряд из пяти гостей. Для этого ряда $g = [3, 1, 0, 2, 4]$ и $s = [2, 1, 3, 0, 4]$.

Приложение работает следующим образом:

- Дорка вводит числа I, J, K — номера трех различных гостей.
- Приложение сообщает ей минимальное количество гостей, которые будут видны, если все три выбранных гостя попадут на фотографию. Формально приложение выведет значение $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$.

Например, рассмотрим ситуацию, показанную на Рис. 1:

- Гости $I = 0$, $J = 2$ и $K = 4$ сидят на местах $s_I = 2$, $s_J = 3$ и $s_K = 4$. Если Дорка выберет их, приложение выведет значение $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$.

  Иными словами, самая узкая фотография, содержащая гостей 0, 2 и 4, включает в себя только этих трех гостей.

- Гости $I = 0$, $J = 4$ и $K = 3$ сидят на местах $s_I = 2$, $s_J = 4$ и $s_K = 0$. Если Дорка выберет их, приложение выведет значение $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$.

  Иными словами, фотография, содержащая этих трех гостей, должна содержать всех 5 гостей.

Помогите Веронике определить правильный порядок рассадки с помощью приложения Дорки. А именно, ваша программа должна определить и вывести последовательность $g_0, g_1, ..., g_{N-1}$. Всегда

существует ровно два правильных ответа (один является обратным другому), и вы можете вывести любой из них. Ваш балл будет зависеть от количества запросов, которые сделает ваше решение.

Реализация

⇨ Это интерактивная задача. Ваша программа будет использовать стандартный ввод и вывод для взаимодействия с проверяющей системой в формате, описанном ниже.

Ваша программа должна начать с чтения одной строки ввода, содержащей целое положительное число T — количество тестовых примеров.

Для каждого тестового примера ваша программа должна начать с чтения одной строки ввода, содержащей целое положительное число N — количество мест, которое также является количеством гостей.

Чтобы сделать запрос, ваша программа должна вывести строку вида «? I J K», где $0 \le I, J, K \le N-1$ — три **различных** числа.

После выполнения запроса ваша программа должна прочитать одну строку, содержащую одно целое положительное число — ответ на ваш запрос.

Чтобы дать ответ с правильным порядком рассадки, ваша программа должна вывести строку вида «! g_0 … g_{N-1}».

После прохождения всех T тестовых примеров программа должна нормально завершиться.

Обратите внимание, что официальная проверяющая система, используемая в CMS для тестирования вашего решения, может быть **адаптивной**. Это означает, что для некоторых тестовых примеров перестановка гостей не определена заранее. Вместо этого проверяющая система может решать, какую из оставшихся перестановок использовать, в зависимости от запросов, уже заданных вашей программой.

Сброс буфера. Если вы не используете предоставленные шаблоны, обязательно очищайте стандартный вывод после вывода каждой строки, иначе ваше решение может быть оценено как *Неверное (Not correct)*. В Python это происходит автоматически, если вы используете `input()` для чтения строк, и вы можете использовать `print(..., flush=True)`, чтобы принудительно очистить буфер. В C++ команда `cout << endl;` очищает буфер в дополнение к выводу символа новой строки; при использовании printf используйте `fflush(stdout)`.

Ограничения

- $1 \le T \le 10$.
- N будет равно 5 (только в примере), 8, 40 или 2000.
- Для каждого тестового примера вы можете сделать не более 10 000 запросов.

Система оценки

Ваша программа будет протестирована на нескольких наборах тестов, разбитых на подзадачи. Чтобы получить баллы за подзадачу, нужно чтобы ваша программа прошла все тесты в ней.

- **Подзадача 0 [0 баллов]**: Пример ($N = 5$).
- **Подзадача 1 [9 баллов]**: $N = 8$.
- **Подзадача 2 [11 баллов]**: $N = 2000$, гости 0 и 1 сидят рядом.
- **Подзадача 3 [15 баллов]**: $N = 40$.
- **Подзадача 4 [65 баллов]**: $N = 2000$.

Для подзадач 1 и 2 любое решение, в котором проходит все тесты, получит полный балл.

Для подзадач 3 и 4 ваше решение должно правильно пройти все тесты, чтобы получить хоть какие-то баллы, а ваш балл будет зависеть от Q_s — наибольшего количества запросов, которое потребовалось вашему решению для прохождения теста. Пусть $X_s = \max(1, Q_s/N)$. Баллы за подзадачи 3 и 4 вычисляются следующим образом:

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

Значение score_s округляется до ближайшего целого числа для каждой подзадачи, а ваш итоговый балл является суммой этих значений. Чтобы получить полный балл, вам нужно решить подзадачу 3 максимум за 55 запросов, а подзадачу 4 — максимум за 2597 запросов. Примеры значений Q_s и баллов для подзадач 3 и 4 приведены ниже.

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

Примеры ввода/вывода

Грейдер	Решение
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Пояснение

В примере входных данных содержится один тестовый пример ($T = 1$) с $N = 5$ гостями. Скрытая конфигурация гостей в этом примере соответствует Рис. 1.

Первый запрос, сделанный примером решения, — 0, 2, 4. Ответ 3 на этот запрос говорит нам, что эти гости сидят, в некотором неизвестном порядке, на трех соседних местах.

Ответ 3 на второй запрос говорит нам то же самое о гостях 3, 0 и 1.

Теперь мы можем сделать вывод, что гость 0 должен сидеть посередине, а гости 2 и 4 — с одной стороны, а гости 1 и 3 — с другой стороны.

После третьего запроса мы уже можем быть уверены, что гости должны сидеть либо в порядке $[3, 1, 0, 2, 4]$, либо в обратном порядке $[4, 2, 0, 1, 3]$. Мы можем вывести любой из этих порядков.

Шаблоны кода и подробности оценивания в CMS

Мы настоятельно рекомендуем использовать предоставленные шаблоны кода для C++ и Python. Они проверяют, успешно ли прошло взаимодействие с проверяющей системой, и корректно завершают работу, если это не так.

Если вы не используете предоставленные шаблоны, в случаях, когда ваше решение неверно, CMS может отобразить неправильный вердикт. Например, вместо «Output isn't correct» вы можете получить «Execution killed by signal» или «Execution timed out (wall clock limit exceeded)».

Мы также рекомендуем использовать инструмент тестирования (см. ниже) для проверки вашего решения локально перед отправкой. Инструмент тестирования проверяет выходные данные вашего решения и сообщает об использовании недопустимых запросов.

Инструмент тестирования

Чтобы упростить тестирование вашего решения, мы предоставляем простой инструмент, который вы можете скачать из CMS. Использование инструмента не является обязательным. Обратите внимание, что официальная проверяющая система в CMS отличается от инструмента тестирования.

Для использования инструмента вам понадобится файл входных данных. Вы можете использовать предоставленный пример входных данных `seatingplan.input0.txt` или создать свой собственный. Файл ввода должен начинаться со строки, содержащей количество T тестовых примеров, а затем для каждого тестового примера должны идти две строки: одна строка с числом N и затем одна строка с номерами $g_0, g_1, ..., g_{N-1}$.

Для программ на Python, например `seatingplan.py` (обычно запускаемых как `pypy3 seatingplan.py`), запускайте инструмент тестирования следующим образом:

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

Для программ на C++, сначала скомпилируйте свое решение:

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

а затем запустите инструмент тестирования:

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