

Існує багато легенд про Пізанську вежу в Торуні. Стіна вежі є колом із  $N \geq 3$  рівновіддаленими дверима (тобто, двері — це вершини правильного  $N$ -кутника). Двері пронумеровані від 0 до  $N - 1$ , але у **випадковому порядку**. Більше деталей у розділі про оцінювання.

Одна з менш відомих легенд розповідає, що кожен новий мешканець вежі мав пройти певне випробування. Його мета — перерахувати двері, починаючи з деяких дверей і рухаючись по колу (за або проти годинникової стрілки), відвідуючи кожні двері рівно один раз.

Це потрібно зробити, не бачачи самої вежі. Натомість мешканець може ставити питання такого типу: «Дано трое різних дверей  $x, y, z$ , між якими з них відстань найменша:  $\{x, y\}$ ,  $\{y, z\}$  чи  $\{z, x\}$ ?». Відповіддю буде список усіх пар (із наведених трьох), що мають найменшу евклідову відстань — довжину найкоротшого відрізка між дверима.

Ваше завдання — написати програму, яка за мінімальну кількість запитів визначить порядок дверей.

## Формат взаємодії

Це інтерактивна задача. Ваша програма має взаємодіяти з інтерактором через стандартний потік входу та виходу.

На початку програма зчитує два числа  $t$  та  $k$  ( $1 \leq t \leq 100$ ,  $1 \leq k \leq 12\,000$ ), що означають кількість тестів та максимальну дозволenu середню кількість запитів відповідно.

Для кожного тесту програма спочатку читає число  $n$  ( $3 \leq n \leq 500$ ) — кількість дверей у вежі.

Далі програма може робити запити у такому форматі:

- Рядок у вигляді  
 $? \ x \ y \ z$   
де  $x, y, z$  — різні цілі числа ( $0 \leq x, y, z \leq n - 1$ ). Цей рядок задає одне питання щодо дверей  $x, y$  і  $z$ .
- У відповідь інтерактор повертає:  
 $r$   
 $a_1 \ b_1$   
 $\dots$   
 $a_r \ b_r$   
де  $r$  ( $1 \leq r \leq 3$ ) — кількість пар дверей з найменшою відстанню між ними. Кожна пара представлена числами  $a_i, b_i$  ( $a_i, b_i \in \{x, y, z\}$ , і  $a_i < b_i$ ).

Коли ви визначили порядок дверей, виведіть:

$! \ x_0 \ x_1 \ \dots \ x_{n-1}$

де  $x_0, x_1, \dots, x_{n-1}$  — порядок дверей. Можна починати з будь-яких дверей та йти в будь-якому напрямку. (Зверніть увагу що існує  $2n$  правильних відповідей в залежності від початкових дверей і напрямку руху. Будь-яка з них буде зарахована.)

**Не забудьте очищати буфер виводу після кожного запиту або відповіді, використовуючи `cout.flush()` (або `fflush(stdout)` якщо використовуєте `printf`) в C++ або `sys.stdout.flush()` в Python. Інакше ваше рішення може отримати Time Limit Exceeded.**

Після відповіді перейдіть до наступного набору або завершіть програму якщо всі набори були оброблені.

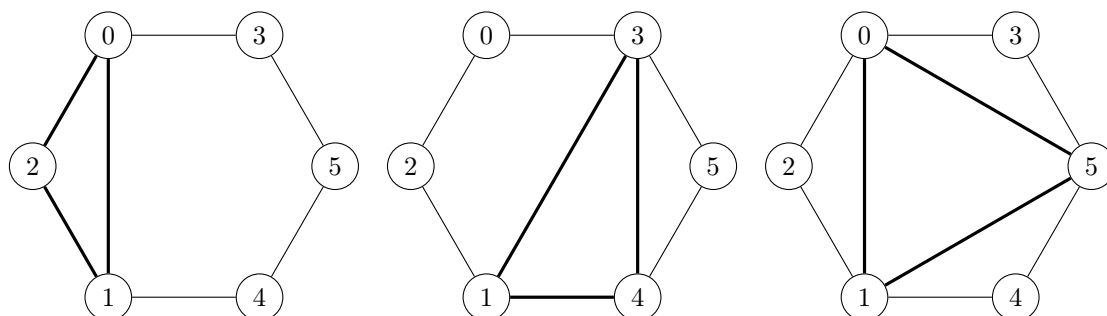
Ваша програма не може відкривати ніякі файли чи використовувати будь-які інші ресурси. Можна використовувати стандартний потік помилок (standart error stream) для дебагу, але враховуйте, що запис на цей потік вимагає часу.

Інтерактор не є адаптивним: порядок дверей для кожного тесту фіксований і не змінюється.

## Приклад взаємодії

Припустимо, що є один тест з  $n = 6$ , а порядок дверей: 5, 3, 0, 2, 1, 4. Взаємодія може виглядати так:

Інтерактор	Ваша програма	Коментар
1 100		$t = 1, k = 100$ .
6		Інтерактор надає кількість дверей.
	? 0 1 2	Запит на найближчу пару серед 0, 1, 2.
2 0 2 1 2		Найближчі — $\{0, 2\}$ і $\{1, 2\}$ .
	? 4 1 3	Наступний запит.
1 1 4		Найближча пара — $\{1, 4\}$ .
	? 0 5 1	Ще один запит.
3 0 5 0 1 1 5		Всі пари однаково близькі.
	! 4 5 3 0 2 1	Ваша програма виводить правильний порядок дверей.



**Пояснення до прикладу:** На малюнках зображено порядок дверей та трикутники, відповідно до трьох запитів. У першому — видно, що  $\{0, 2\}$  і  $\{1, 2\}$  найкоротші. У другому —  $\{1, 4\}$  найкоротша. У третьому — всі однаково близькі.

Правильні відповіді також: 0, 2, 1, 4, 5, 3 або 5, 4, 1, 2, 0, 3 (та інші варіанти).

## Оцінювання

Оцінювання розбите на підзадачі. Для кожної підзадачі є один тест, що містить рівно  $t = 100$  наборів вхідних даних.

Обчислюється середня кількість запитів  $k^*$  на набір (кількість запитів поділена на кількість наборів даних). Якщо  $k^* > k$  — отримаєте 0 балів за підзадачу. Інакше — повний бал (крім останньої підзадачі).

Для останньої підзадачі бали обчислюються так:

$$\left\lceil 56 \cdot \min \left( 1, \frac{12000 - k^*}{7800} \right) \right\rceil$$

що означає, що ваш бал збільшується лінійно з 0 до 56 коли  $k^*$  зменшується з 12000 до 4200.

Зверніть увагу, що якщо відповідь на хоча б один набір неправильна то ви отримаєте 0 балів за всю підзадачу незалежно від кількості запитів.

Додаткові обмеження по підзадачам наведені нижче.

Підзадача	Обмеження	Бали
1	$k = 8000, 3 \leq n \leq 9$	6
2	$k = 4500, 40 \leq n \leq 50$	7
3	$k = 3000, 90 \leq n \leq 100$	9
4	$k = 4500, n = 400$ , існує правильна відповідь $x_0, \dots, x_{n-1}$ , де $x_i = i$ для $200 \leq i \leq 399$	22
5	$k = 12000, n = 500$	до 56

Кожен тест генерується **випадково**:  $n$  вибирається **рівномірно** (кожен  $n$  однаково ймовірне) в межах обмежень на підзадачі, порядок дверей — теж **випадково рівномірно** (кожен порядок однаково ймовірний) в межах обмежень на підзадачі.