Toy Design

Задача	ToyDesign
Вхідні дані	Інтерактивна задача
Вихідні дані	Інтерактивна задача
Ліміт часу	1 секунда
Ліміт пам'яті	256 МБ

Ви працюєте в компанії, яка розробляє іграшки. Нова іграшка, що зараз створюється вами, працює так: n контактів, пронумерованих від 1 до n, стирчать із коробки. Деякі пари контактів з'єднані дротами всередині коробки. (Іншими словами, контакти та дроти утворюють неорієнтований граф, де контакти є вершинами, а дроти — ребрами.) Дроти не видно ззовні, і єдиний спосіб дізнатися щось про них — використовувати **тестер** на контактах: ми можемо вибрати два контакти i і j так, що $i \neq j$, і тестер покаже, чи ці два контакти з'єднані всередині коробки, прямо чи опосередковано (за допомогою інших контактів). (Таким чином, тестер повідомляє, чи є між цими контактами шлях на графі.)

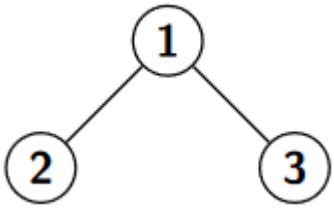
Ми будемо називати набір з'єднань всередині коробки **планом** іграшки.

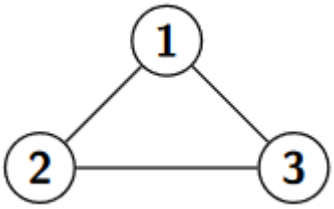
Ви використовуєте спеціалізоване програмне забезпечення для роботи з цими планами. Це програмне забезпечення працює так: воно починається з певного плану іграшки, який ми позначаємо як «план 0». Програма не показує з'єднання всередині коробки. Замість цього ви можете кілька разів виконати таку операцію:

- Ви обираєте план номер a і два номери контактів i і j, де $i \neq j$.
- Програмне забезпечення повідомляє вам, що станеться, якщо ми використаємо тестер на цих двох контактах. Іншими словами, він повідомляє вам, чи контакти i та j з'єднані (прямо чи опосередковано) у плані a.
- Крім того, якщо контакти не були з'єднані в плані a, тоді створюється новий план, який має всі з'єднання з плану a плюс одне додаткове пряме з'єднання між i і j. Йому надається наступний доступний номер. (Отже, перший план, створений таким чином, матиме номер 1, потім номер 2 і так далі.) Зверніть увагу, що це не змінює план a, лише створює новий, який має додаткове з'єднання.

Ваша мета — дізнатися якомога більше про план 0 за допомогою цієї операції.

Зауважте, що не завжди можливо визначити точний набір з'єднань для плану 0, оскільки немає способу розрізнити прямі та опосередковані з'єднання. Наприклад, розглянемо наступні два плани з $n = 3$:





Для будь-якої пари контактів, тестер повідомить те, що вони з'єднані, тому ми не зможемо розрізнити їх за допомогою програмного забезпечення, описаного вище.

Ваша мета — визначити будь-який план, еквівалентний плану 0. Два плани є **еквівалентними**, якщо для всіх пар контактів, тестер повідомляє одинакову відповідь для обох планів.

Реалізація

Це інтерактивна задача. Ви повинні реалізувати функцію

```
void ToyDesign(int n, int max_ops);
```

яка визначає план, *еквівалентний* плану 0. Для взаємодії ви можете використовувати дві функції. Перша з них:

```
int Connected(int a, int i, int j);
```

де $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$ і a не повинно перевищувати кількість планів, створених на даний момент. Якщо контакти i і j (прямо чи опосередковано) з'єднані в плані a, то функція поверне число a. В іншому випадку вона поверне кількість планів, створених на даний момент, плюс один, яка стане номером, призначеним новому плану, який має всі зв'язки плану a плюс зв'язок між i і j. Функцію `Connected` можна викликати щонайбільше `max_ops` разів.

Коли ваша програма завершить роботу з функціями `Connected`, вона має описати план, еквівалентний плану 0. Щоб описати план, програма повинна викликати:

```
void DescribeDesign(std::vector<std::pair<int,int>> result);
```

Параметр `result` — це вектор пар цілих чисел, що описує прямі з'єднання між контактами. Кожна пара відповідає одному з'єднанню та має містити два номери контактів з'єднання. Між

кожною (невпорядкованою) парою контактів має бути щонайбільше один прямий зв'язок, і не має бути прямих зв'язків між одним і тим самим контактом. Виклик цієї функції зупиняє виконання програми.

Обмеження

- $2 \leq n \leq 200$

Оцінювання

- Підзадача 1 (10 балів): $n \leq 200$, *max_ops* $= 20\,000$
- Підзадача 2 (20 балів): $n \leq 8$, *max_ops* $= 20$
- Підзадача 3 (35 балів): $n \leq 200$, *max_ops* $= 2\,000$
- Підзадача 4 (35 балів): $n \leq 200$, *max_ops* $= 1\,350$

Зразок взаємодії

Дія учасника	Дія грейдера	Пояснення
	`ToyDesign(4, 20)`	В іграшці 4 контакти. Вам потрібно визначити будь-який план, еквівалентний плану 0, викликавши `Connected` щонайбільше 20 разів.
`Connected(0, 1, 2)`	`Повертає 1.`	Контакти 1 і 2 не з'єднані ніяк в плані 0. Створюється новий план 1.
`Connected(1, 3, 2)`	`Повертає 2.`	Контакти 3 і 2 не з'єднані ніяк в плані 1. Створюється новий план 2.
`Connected(0, 3, 4)`	`Повертає 0.`	Контакти 3 і 4 з'єднані прямо або опосередковано в плані 0. Новий план не створюється.
`DescribeDesign({{3, 4}})`	-	Ми описуємо план, який має лише одне з'єднання: контакти 3 і 4.

Зразок грейдера

Наданий зразок грейдера, `grader.cpp`, у вкладенні задачі `ToyDesign.zip`. Грейдер зчитує вхідні дані зі стандартного вводу в такому форматі:

- Перший рядок містить кількість контактів, n, кількість прямих з'єднань, m і *max_ops*
- Наступні m рядків містять прямі з'єднання як пари контактів.

Зразок грейдера зчитує вхідні дані та викликає функцію `ToyDesign` у вашому рішенні. Грейдер виведе одне з наступних повідомлень, залежно від поведінки вашого рішення:

- "Wrong answer: Number of operations exceeds the limit" («Неправильна відповідь: кількість операцій перевищує ліміт»), якщо кількість викликів до `Connected` перевищує *max_ops*
- "Wrong answer: Wrong design id" («Неправильна відповідь: неправильний номер плану»), якщо параметр a під час виклику `Connected` є номером плану, якого не існує на момент виклику.
- "Wrong answer: Incorrect design" («Неправильна відповідь: неправильний план»), якщо план, описаний за допомогою `DescribeDesign`, не еквівалентний план 0.
- "OK!" («Гаразд!») якщо план, описаний за допомогою `DescribeDesign`, еквівалентний плану 0.

Щоб скомпілювати зразок грейдера з вашим рішенням, ви можете використати наступну команду в терміналі:

```
g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp
```

де `solution.cpp` — ваш файл рішення, який потрібно надсилати до CMS. Щоб запустити програму із зразком вхідних даних, наданим у вкладенні, введіть таку команду в терміналі:

```
./solution < input.txt
```