Toy Design

Problem Name	ToyDesign
Input File	Interactive Task
Output File	Interactive Task
Time limit	1 second
Memory limit	256 megabytes

Вы работаете в компании, производящей игрушки. Новая создаваемая игрушка работает так: Есть n контактов, пронумерованных от 1 до n, выступающих из коробки. Некоторые пары контактов соединены проводами внутри коробки. (Другими словами, контакты и провода образуют неориентированный граф, где контакты это вершины, а провода это рёбра). Провода не видны снаружи, и единственным способом узнать что-то о них является использование **тестера** на паре контактов: Мы можем выбрать контакты i и j, такие что $i \neq j$, и тестер покажет, соединены ли контакты внутри коробки, прямо или косвенно. (То есть тестер показывает, существует ли путь между этими контактами в графе).

Будем назвать набор соединений внутри коробки **дизайном** игрушки.

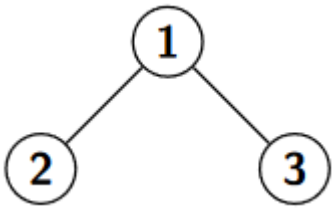
Вы используете специальное программное обеспечение для тестирования и создания таких дизайнов. Это программное обеспечение работает так: Оно начинает с некоторого дизайна игрушки, который мы назовем "дизайн 0". Оно не показывает вам соединений внутри коробки. Вместо этого вы можете несколько раз выполнить следующую операцию, состоящую из трех шагов:

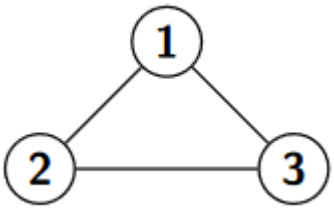
1. Вы выбираете дизайн с номером a и два контакта с номерами i и j, такими что $i \neq j$.
2. Программное обеспечение показывает вам, что случится, если мы используем тестер для этих контактов. Другими словами, оно показывает, соединены ли контакты i и j (прямо или косвенно) в дизайне a.
3. Также, если контакты не были соединены прямо или косвенно в дизайне a, то оно создаёт новый дизайн, который включает все соединения из дизайна a плюс одно дополнительное прямое соединение между i и j. Этот дизайн получает следующий доступный номер дизайна. (Так, первый дизайн созданный таким образом будет иметь

номер 1, затем номер 2, и так далее). Заметим, что при этом не меняется дизайн a, просто создаётся новый дизайн с дополнительным соединением.

Ваша цель узнать как можно больше о дизайне 0, используя эту операцию.

Заметим, что не всегда возможно установить точный набор соединений для дизайна 0, так как нет возможности различить прямое или косвенное соединение. Например, рассмотрим следующие два дизайна с $n = 3$:





Тестер покажет, что соединены любые пары контактов для обоих дизайнов, так что мы не в состоянии различить их, используя программное обеспечение, описанное выше.

Ваша цель определить любой дизайн, эквивалентный дизайну 0. Два дизайна **эквивалентны**, если тестер покажет одинаковый результат для любой пары контактов обоих дизайнов.

Implementation

Это интерактивная задача. Вы должны реализовать функцию

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

которая определяет дизайн, *эквивалентный* дизайну 0. Ваша реализация должна достигнуть этой цели, совершая две определённые операции ноль или более раз. В одной операции, ваша программа должна вызвать функцию:

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

где $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, и a не должно превышать количества дизайнов, созданных до сих пор. Если контакты i и j соединены (прямо или косвенно) в дизайне a, функция вернёт значение a. Иначе, она вернёт количество дизайнов, созданных до сих пор, плюс один, которое станет номером, назначенным новому дизайну, имеющему все соединения дизайна a плюс соединение между i и j. Функция `Connected` может быть вызвана не более `max_ops` раз.

Когда ваша программа закончит с вызовами `Connected`, она должна описать дизайн, эквивалентный дизайну 0. Чтобы описать дизайн, программа должна вызвать:

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

Параметр `result` это вектор пар целых чисел, описывающий прямые соединения между контактами. Каждая пара соотвветсвует одному соединению и должна содержать два номера соединяемых контактов. Должно быть не более одного прямого соединения между любой (неупорядоченной) парой контактов, не должно быть прямых соединений контакта самого с собой. Вызов этой функции завершит выполнение программы.

Constraints

- $2 \leq n \leq 200$

Scoring

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

Sample Interaction

Действия вашей программы	Действия грейдера	Пояснения
	`ToyDesign(4, 20)`	У игрушки 4 контакта. Вам необходимо найти дизайн, эквивалентный дизайну 0, сделав не более 20 вызовов `Connected`.
`Connected(0, 1, 2)`	`Возвращает 1.`	Контакты 1 и 2 не связаны прямо или косвенно в дизайне 0. Новый дизайн 1 создан.
`Connected(1, 3, 2)`	`Возвращает 2.`	Контакты 3 и 2 не связаны прямо или косвенно в дизайне 1. Новый дизайн 2 создан.
`Connected(0, 3, 4)`	`Возвращает 0.`	Контакты 3 и 4 связаны явно или неявно и дизайне 0. Новый дизайн не создается.
`DescribeDesign({{3, 4}})`	-	Описывается дизайн с одним соединением: контакты 3 и 4.

Sample Grader

Предоставляемый пример грейдера, `grader.cpp`, в приложении к задаче `ToyDesign.zip`, читает данные из стандартного ввода в следующем формате:

- Первая строка содержит количество контактов n, количество прямых соединений m и *max_ops*.
- Следующие m строк содержат прямые соединения как пары контактов.

Пример грейдера читает входные данные и вызывает функцию `ToyDesign` в решении участника. Грейдер выведет одно из следующих сообщений, в зависимости от результатов работы вашего решения:

- "Wrong answer: Number of operations exceeds the limit.", если число вызовов `Connected` превысит *max_ops*
- "Wrong answer: Wrong design id.", если аргумент a, переданный `Connected` задает номер дизайна, которого в тот момент не существует.
- "Wrong answer: Incorrect design.", если дизайн, переданный `DescribeDesign`, не эквивалентен дизайну 0.
- "OK!", если дизайн, переданный `DescribeDesign` эквивалентен дизайну 0.

Чтобы скомпилировать пример грейдера с вашим решением, вы можете использовать следующую команду в командной строке операционной системы:

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

где `solution.cpp` является именем вашего файла, отсылаемого в CMS. Чтобы запустить программу с примером входных данных, предоставленным во вложении, наберите следующую команду в командной строке операционной системы:

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