Diseño de juguetes

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

Estás trabajando para una compañía que diseña juguetes. Un nuevo juguete que está siendo creado, funciona de la siguiente manera: Tiene n pines, numerados de 1 a n, sobresaliendo de una caja. Algunos pares de pines están conectados con cables adentro de la caja. (En otras palabras, los pines y los cables forman un grafo no dirigido, donde los pines son los vértices y los cables las aristas). Los cables no se ven desde afuera y la única forma de saber algo de ellos es usar un **multímetro** en los pines: Podemos escoger dos pines i y j tal que $i \neq j$ y el multímetro dirá si esos dos pines están conectados adentro de la caja, ya sea de forma directa o indirecta. (El multímetro dirá si hay un camino en el grafo entre esos dos pines).

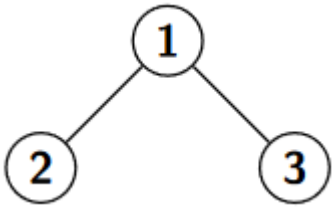
Al conjunto de conexiones adentro de la caja le llamaremos **diseño** del juguete.

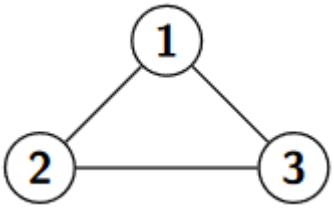
Estás usando un software especializado para consultar y crear estos diseños. Este software funciona de la siguiente manera: Empieza con un diseño del juguete al que llamaremos "diseño 0". No te muestra las conexiones adentro de la caja de este diseño. En cambio puedes ejecutar la siguiente operación de tres pasos repetidamente:

1. Escoges un número de diseño a y dos pines i y j tal que $i \neq j$.
2. El software te dice que pasaría si usas el multímetro en esos dos pines. En otras palabras, te dice si el pin i y el pin j están (directa o indirectamente) conectados en el diseño a.
3. En caso de que los pines no estén directa o indirectamente conectados en el diseño a, crea un nuevo diseño que tiene todas las conexiones del diseño a más una conexión directa adicional entre i y j. A este diseño se le da el siguiente número de diseño disponible. (Entonces, el primer diseño creado de esta forma, tendrá el número 1, el segundo el número 2 y así sucesivamente). Nota que esto no cambia el diseño a, solo crea un nuevo diseño que tiene la conexión adicional.

Tu objetivo es aprender lo más que puedas sobre el diseño 0 usando esta operación.

Nota que no siempre es posible determinar el conjunto exacto de conexiones para el diseño 0, porque no hay forma de distinguir conexiones directas e indirectas. Por ejemplo, considera los siguientes dos diseños con $n = 3$:





El multímetro reportaría cualquier par de pines como conectados para ambos diseños, por lo que no podríamos distinguirlos usando el software previamente mencionado.

Tu objetivo es determinar cualquier diseño que sea equivalente al diseño 0. Dos diseños son **equivalentes** si el multímetro reporta el mismo resultado en ambos diseños para todos los pares de pines.

Implementación

Este es un problema interactivo. Debes implementar una función

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

que determina un diseño que es *equivalente* al diseño 0. Tu implementación debe lograr este objetivo llamando dos funciones como se describe a continuación.

La primer función que puedes llamar es:

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

donde $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, y a no debe exceder el número de diseños creados hasta el momento. Si los pines i y j están conectados (directa o indirectamente) en el diseño a, entonces regresará el valor de a. De lo contrario, regresará el número de diseños creados hasta el momento, más uno, que se convierte en el número asignado al nuevo diseño que tiene las conexiones del diseño a más la conexión entre i y j. La función `Connected` puede ser llamada a lo más `max_ops` veces.

Cuando tu programa termine con las operaciones de `Connected`, debe describir un diseño que sea equivalente al diseño 0. Para describir un diseño, el programa debe llamara a:

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

El parámetro `result` es un vector de pares de enteros que describe las conexiones directas entre los pines. Cada par corresponde a una conexión y debe contener los números de ambos pines que forman la conexión. Debe haber a lo más una conexión directa entre cada pareja de pines, y ningún pin debe tener conexión consigo mismo. Llamar a esta función, termina la ejecución de tu programa.

Límites

- $2 \leq n \leq 200$

Subtareas

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

Interacción de ejemplo

Acción de la concursante	Acción del evaluador	Explicación
	`ToyDesign(4, 20)`	Hay 4 pines en el juguete. Necesitas determinar cualquier diseño que sea equialente al diseño 0 llamando a `Connected` a lo más 20 veces.
`Connected(0, 1, 2)`	`Regresa 1.`	Los pines 1 y 2 no están directa o indirectamente conectados en el diseño 0. Se crea el nuevo diseño 1.
`Connected(1, 3, 2)`	`Regresa 2.`	Los pines 3 y 2 no están directa o indirectamente conectados en el diseño 1. Se crea el nuevo diseño 2.
`Connected(0, 3, 4)`	`Regresa 0.`	Los pines 3 y 4 están directa o indirectamente conectados en el diseño 0. No se crea ningún nuevo diseño.
`DescribeDesign({{3, 4}})`	-	Describimos un diseño que solo tiene una conexión: Los pines 3 y 4.

Evaluador de ejemplo

El evaluador de ejemplo dado, `grader.cpp`, adjunto al problema `ToyDesign.zip`, lee la entrada de la entrada estándar en el siguiente formato:

- La primera línea contiene el número de pines, n, el número de conexiones directas m y *max_ops*
- Las siguientes m líneas contienen las conexiones directas como tuplas de pines.

El evaluador de ejemplo lee la entrada y llama a la función `ToyDesign` de tu solución. El evaluador imprimirá alguno de los siguientes mensajes dependiendo del comportamiento de tu solución:

- "Wrong answer: Number of operations exceeds the limit.", si el número de llamadas a `Connected` excede *max_ops*
- "Wrong answer: Wrong design id.", si el parámetro a en una llamada a `Connected` es el número de un diseño que no existe en el momento en el que se hizo la llamada.
- "Wrong answer: Incorrect design.", si el diseño descrito por `DescribeDesign` no es equivalente al diseño 0.
- "OK!" si el diseño escrito por `DescribeDesign` es equivalente al diseño 0.

Para compilar el evaluador de ejemplo, puedes usar el siguiente comando en la consola:

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

donde `solution.cpp` es el archivo de tu solución que enviarás a CMS. Para ejectura el programa con la entrada del archivo, copia el siguiente comando en la consola:

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