Toy Design

Nombre del problema	ToyDesign
Fichero de entrada	Problema interactivo
Fichero de salida	Problema interactivo
Límite de tiempo	1 segundo
Límite de memoria	256 megabytes

Trabajas en una empresa que diseña juguetes. Un juguete nuevo en fase de diseño funciona de la siguiente manera: Hay n pines, numerados de 1 a n, que salen de una caja. Algunas parejas de pines están conectadas por cables dentro de la caja. (En otras palabras, los pines y los cables forman un grafo no dirigido, donde los pines son vértices y los cables son aristas.) Los cables no se ven desde fuera, y la única forma de obtener información es usar un **multímetro** en los pines: Podemos escoger dos pines i y j tal que $i \neq j$, y el multímetro nos dirá si esos pines están conectados en el interior de la caja, ya sea directa o indirectamente. (Es decir, el múltimetro indica si existe un camino entre esos pines en el grafo.)

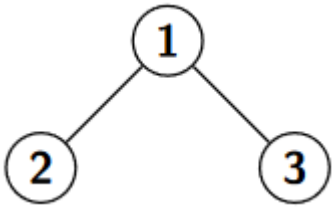
Llamaremos al conjunto de conexiones en el interior de la caja **diseño** del juguete.

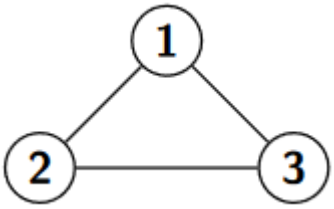
Estás usando software especializado para consultar y crear estos diseños. Este software funciona de la siguiente manera: Empieza con un diseño del juguete que llamamos "diseño 0". No te muestra las conexiones en el interior de la caja de este diseño. En su lugar, puedes llevar a cabo la siguiente operación de tres pasos múltiples veces:

1. Eliges un número de diseño a y dos números de pines i y j tal que $i \neq j$.
2. El software te dice qué pasaría si usaramos el multímetro en esos pines. En otras palabras, te dice si los pines i y j están (directa o indirectamente) conectados en el diseño a.
3. También, si los pines no estaban directa o indirectamente conectados en el diseño a, entonces crea un nuevo diseño que tiene todas las conexiones del diseño a junto a una conexión directa adicional entre i y j. A este diseño se le da el siguiente número de diseño que esté disponible. (Es decir, el primer diseño creado de esta forma tendrá el número 1, el segundo el número 2, etc...) Ten en cuenta que esto no modifica el diseño a, únicamente crea un nuevo diseño que tiene la conexión adicional.

Tu objetivo es averiguar todo lo que puedas sobre el diseño 0 usando esta operación.

Ten en cuenta que no siempre se puede determinar el conjunto de conexiones de manera exacta para el diseño 0, porque no hay manera de diferenciar conexiones directas e indirectas. Por ejemplo, considera los siguientes dos diseños con $n = 3$:





El multímetro nos diría que cualquier pareja está conectada, en ambos diseños, por lo que no podemos distinguirlos usando el software descrito previamente.

Tu objetivo es determinar cualquier diseño que sea equivalente al diseño 0. Dos diseños son **equivalentes** si el multímetro nos daría el mismo resultado en todas las parejas de pines.

Implementación

Este es un problema interactivo. Tienes que implementar la función

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

que determina un diseño *equivalente* al diseño 0. Tu implementación debe lograr este objetivo llamando a dos funciones como se describe a continuación. La primera función a la 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 ser mayor que el número de diseños creados hasta el momento. Si los pines i y j están (directa o indirectamente) conectados en el diseño a, entonces devolverá a. En caso contrario, devolverá el número de diseños creados hasta el momento más uno, que pasará a ser el número asignado al nuevo diseño que tiene todas las conexiones del diseño a junto a la conexión adicional entre i y j. La función `Connected` puede ser llamada como mucho `max_ops` veces.

Cuando tu programa haya acabado con las operaciones `Connected`, debe describir un diseño equivalente al diseño 0. Para describir un diseño, el programa debe llamar a:

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

El parámetro `result` es un vector de pares de enteros describiendo las conexiones directas entre los pines. Cada par se corresponde con una conexión y debe contener los números de ambos

pines que forman la conexión. Debe haber como máximo una conexión directa entre cada pareja (desordenada) de pines, y ningún pin debe tener conexiones a sí mismo. Llamar a esta función termina la ejecución de tu programa.

Restricciones

- $2 \leq n \leq 200$

Puntuación

- 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* $= 1\,350$

Ejemplo de interacción

Acción de la concursante	Acción del corrector	Explicación
	`ToyDesign(4, 5)`	Hay 4 pines en el juguete. Tienes que determinar cualquier diseño equivalente al diseño 0 llamando a `Connected` como mucho 5 veces.
`Connect(0, 1, 2)`	`Devuelve 1.`	Los pines 1 y 2 no están conectados directa o indirectamente en el diseño 0. Se crea el nuevo diseño 1.
`Connected(1, 3, 2)`	`Devuelve 2.`	Los pines 3 y 2 no están conectados directa o indirectamente en el diseño 1. Se crea el nuevo diseño 2.
`Connected(0, 3, 4)`	`Devuelve 0.`	Los pines 3 y 4 están conectados directa o indirectamente en el diseño 0. No se crea ningún diseño nuevo.
`DescribeDesign({{3, 4}})`	-	Describimos un diseño que solo tiene una conexión: Los pines 3 y 4.

Corrector de ejemplo

El corrector de ejemplo dado, `grader.cpp`, en el archivo adjunto `ToyDesign.zip`, lee el input 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 corrector de ejemplo lee el input y llama la función `ToyDesign` de tu solución.

Para compilar el corrector de ejemplo junto a tu solución, puedes usar el siguiente comando en la terminal:

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

donde `solution.cpp` es el archivo de tu solución que enviarías a CMS. Para ejecutar le programa con el corrector de ejemplo, ejecuta el siguiente comando en la terminal:

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