Toy Design

Nom du problème	ToyDesign
Fichier d'entrée	Tâche interactive
Fichier de sortie	Tâche interactive
Limite de temps	1 seconde
Limite de mémoire	256 megaoctets

Vous travaillez pour une société qui invente des jouets. Un nouveau jouet en train d'être inventé fonctionne comme ceci : il y a n plots, numérotés de 1 à n, qui traversent le couvercle d'une boîte. Certaines paires de plots sont reliées par des fils conducteurs à l'intérieur de la boîte. En d'autres mots, les plots et les fils forment un graphe non orienté, où les plots sont les noeuds et les fils sont les arêtes. Les fils ne sont pas visibles de l'extérieur de la boîte, et la seule manière d'apprendre quelque chose à leur sujet est d'utiliser un **testeur** sur les plots : on peut choisir deux plots i et j tels que $i \neq j$, et le testeur indiquera si ces deux plots sont reliés à l'intérieur de la boîte, que cela soit directement ou indirectement. (Ainsi, le testeur indique s'il y a un chemin entre ces deux plots dans le graphe.)

On appellera **schéma**, l'ensemble des connexions à l'intérieur de la boîte du jouet.

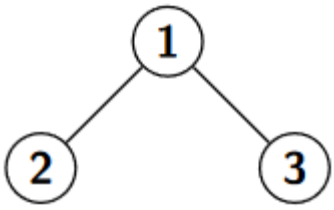
Vous utilisez un logiciel spécialisé pour créer ces schémas et les interroger. Ce logiciel fonctionne comme suit : il commence par un schéma du jouet que l'on appellera "schéma 0". Il ne vous montre pas les connexions à l'intérieur de la boîte pour ce schéma. Par contre, vous pouvez effectuer une ou plusieurs fois l'opération suivante à trois étapes :

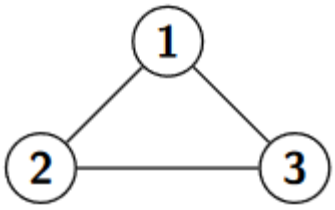
1. Vous choisissez un numéro de schéma a et deux numéros de plots i et j tels que $i \neq j$.
2. Le logiciel vous indique ce qui se produirait si l'on utilisait le testeur sur ces deux plots. Autrement dit, ils vous dit si les plots i et j sont reliés (directement ou indirectement) dans le schéma a.
3. Aussi, si les plots n'étaient pas directement ou indirectement reliés dans le schéma a, alors il crée un nouveau schéma qui a toutes les connexions du schéma a plus une connexion directe entre i et j. Le prochain numéro de schéma est alors attribué à ce nouveau schéma. (Le premier schéma créé de cette manière aura le numéro 1, puis le numéro 2, et ainsi de

suite.) Notez que cela ne change pas le schéma a, cela crée seulement un nouveau schéma qui a la connexion supplémentaire.

Votre but est d'en apprendre le plus possible sur le schéma 0 en utilisant cette opération.

Notez qu'il n'est pas toujours possible de déterminer exactement l'ensemble des connexions du schéma 0, car il n'y a pas de manière de différencier les connexions directes et indirectes. Considérez par exemple les deux schémas suivants, avec $n = 3$:





Le testeur répondrait que toute paire de plots est reliée dans les deux schémas, donc il n'est pas possible de les différencier en utilisant le logiciel décrit précédemment.

Votre objectif est d'identifier un quelconque schéma qui est équivalent au schéma 0. Deux schémas sont **équivalents** si le testeur donne la même réponse dans les deux schémas, pour toute paire de plots.

Implémentation

Ceci est un problème interactif. Vous devez implémenter une fonction

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

qui identifie un schéma qui est *équivalent* au schéma 0. Votre implémentation doit atteindre cet objectif en appelant les deux fonctions décrites ci-dessous. La première fonction que vous pouvez appeler est :

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

où $1 \leq i, j \leq n, i \neq j, a \geq 0$, et a ne doivent pas dépasser le nombre de schémas créés jusque-là. Si les plots i et j sont reliés (directement ou indirectement) dans le schéma a, alors elle retournera a. Autrement, elle retournera le nombre de schémas créés jusqu'à présent, plus un, qui devient le nombre affecté au nouveau schéma qui a toutes les connexions du schéma a plus la connexion entre i et j. La fonction `Connected` peut être appelée au maximum `max_ops` fois.

Quand votre programme a terminé ses opérations `Connected`, il doit décrire un schéma qui est équivalent au schéma 0. Pour décrire un schéma, un programme doit appeler :

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

Le paramètre `result` est un vecteur de paires d'entiers décrivant les connexions directes entre les plots. Chaque paire correspond à une connexion et doit contenir les deux numéros de plots de la connexion. Il doit y avoir au maximum une connexion directe entre chaque paire (non ordonnée) de plots, et aucune connexion directe entre un plot et lui-même. Appeler cette fonction termine l'exécution de votre programme.

Contraintes

- $2 \leq n \leq 200$

Score

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

Exemple d'interaction

Action de la candidate	Action de l'évaluateur	Explication
	`ToyDesign(4, 20)`	Il y a 4 plots dans le jouet. Vous devez identifier un schéma qui est équivalent au schéma 0 en appelant `Connected` au maximum 20 fois.
`Connected(0, 1, 2)`	`Retourne 1.`	Les plots 1 et 2 ne sont pas reliés directement ou indirectement dans le schéma 0. Le nouveau schéma 1 est créé.
`Connected(1, 3, 2)`	`Retourne 2.`	Les plots 3 et 2 ne sont pas reliés directement ou indirectement dans le schéma 1. Le nouveau schéma 2 est créé.
`Connected(0, 3, 4)`	`Retourne 0.`	Les plots 3 et 4 sont reliés directement ou indirectement dans le schéma 0. Aucun nouveau schéma n'est créé.
`DescribeDesign({{3, 4}})`	-	On décrit un schéma qui n'a qu'une connexion : les plots 3

		et 4.

Exemple d'évaluateur

L'exemple d'évaluateur fourni, `grader.cpp`, dans le dossier joint `ToyDesign.zip`, lit l'entrée depuis l'entrée standard dans le format suivant :

- La première ligne contient le nombre de plots n, le nombre de connexions directes m et *max_ops*
- Les m lignes suivantes contiennent les connexions directes données sous forme de couples de plots.

L'exemple d'évaluateur lit l'entrée et appelle la fonction `ToyDesign` de la solution de la candidate. L'évaluateur va donner comme sortie l'un des messages ci-dessous, selon le comportement de votre solution :

- "Wrong answer: Number of operations exceeds the limit.", si le nombre d'appels à `Connected` dépasse *max_ops*
- "Wrong answer: Wrong design id.", si le paramètre a d'un appel à `Connected` est le numéro d'un schéma qui n'existe pas au moment où l'appel est fait.
- "Wrong answer: Incorrect design.", si le schéma décrit avec `DescribeDesign` n'est pas équivalent au schéma 0.
- "OK!" si le schéma décrit avec `DescribeDesign` est équivalent au schéma 0.

Pour compiler l'exemple d'évaluateur avec votre solution, vous pouvez utiliser la commande suivante dans votre terminal :

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

où `solution.cpp` est votre solution à soumettre à CMS. Pour lancer le programme avec l'exemple d'entrée fourni dans le dossier joint, tapez la commande suivante dans votre terminal :

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