Toy Design

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

Nešpecifikovaným spôsobom sme získali fakt zaujímavú hračku: škatuľku s kolíkmi.

Z vnútra škatuľky trčí von n kovových kolíkov. Kolíky majú čísla od 1 po n. Vo vnútri škatuľky sú niektoré dvojice kolíkov prepojené kusmi drôtu. Kolíky a drôty v škatuľke teda tvoria neorientovaný graf. Ty tento graf samozrejme nevidíš.

Dva kolíky sú vodivo prepojené, ak vie z jedného na druhý tiecť elektrický prúd -- buď priamo (drôtom) alebo nepriamo (cez iné kolíky postupnosťou viacerých po sebe idúcich drôtov). Inými slovami, kolíky sú vodivo prepojené, ak medzi nimi v grafe existuje cesta.

Jediný nástroj, ktorým vieš zistiť niečo o obsahu škatuľky, je **magický tester**. Tester sa používa tak, že ho pripojíš na dva kolíky a on ti povie, či sú vo vnútri škatuľky vodivo prepojené alebo nie.

Ale pozor, magický tester má ešte jeden vedľajší účinok!

Škatuľku, ktorú máš na začiatku, budeme volať škatuľka 0.

Ak otestuješ dva kolíky nejakej škatuľky a tester ti povie "áno" (zvolené dva kolíky sú v danej škatuľke prepojené), nič iné sa nestane.

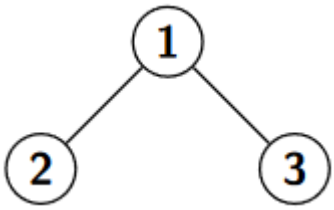
Ale ak dá tester odpoveď "nie", magicky vznikne **nová škatuľka**. Tá dostane prvé nepoužité prirodzené číslo (takže postupne budú vznikať škatuľky 1, 2, 3, ...). Nová škatuľka bude presnou kópiou tej, ktorú sme práve otestovali, až na jeden rozdiel: v novej škatuľke bude navyše existovať aj drôt, ktorý priamo spája tie dva kolíky, ktoré sme práve testovali.

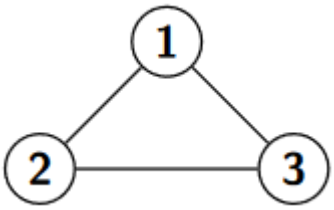
Upozorňujeme, že pôvodná škatuľka (tá, ktorú sme testovali) ostane nezmenená.

Vo všeobecnosti testerom nevieme rozlíšiť, či sú dva kolíky prepojené priamo alebo nepriamo. Dve škatuľky preto budeme volať **ekvivalentné**, ak pre každé dva kolíky jednej by tester dal rovnakú

odpoveď ako pre tie isté dva kolíky druhej z nich.

Napríklad uvažujme nasledovné dve škatuľky pre $n = 3$:





Pre každú z nich by sme pre všetky dvojice kolíkov dostali odpoveď "áno", takže len použitím testera nevieme zistiť, ktorú z nich máme. Tieto dve škatuľky sú teda ekvivalentné.

Tvojou úlohou je zistiť o škatuľke 0 všetko, čo sa pomocou testera dá zistiť. Formálne, tvojou úlohou je povedať, ako vnútri vyzerá ľubovoľná jedna škatuľka, ktorá je ekvivalentná so škatuľkou 0.

Implementation

Toto je interaktívna úloha.

Implementuj nasledovnú funkciu:

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

Táto funkcia má zostrojiť škatuľku ekvivalentnú s tebe neznámou škatuľkou 0. Toto bude prebiehať v dvoch fázach: najskôr môžeš používať magický tester a potom musíš graderu oznámiť zoznam drôtov, ktoré tvoria škatuľku ekvivalentnú so škatuľkou 0.

V prvej fáze môže tvoje riešenie dokola volať nasledovnú funkciu gradera:

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

V každej podúlohe existuje limit, označený `max_ops`. Tento hovorí, najviac koľkokrát môžeš zavolať funkciu `Connected`. Všimni si, že tento limit dostaneš pri každom teste ako druhý parameter funkcie `ToyDesign`.

Zavolaním funkcie `Connected` použiješ magický tester na škatuľku a, a na nej na kolíky i a j. Pre každé volanie tejto funkcie musí platiť:

- $a \geq 0$
- a je číslo nejakej v danej chvíli existujúcej škatuľky
- $1 \leq i, j \leq n$
- $i \neq j$

Ak sú v škatuľke a kolíky i a j vodivo spojené, táto funkcia bude mať návratovú hodnotu a. Táto návratová hodnota znamená, že magický tester odpovedal "áno" a nič iné sa nestalo.

Ak dotyčné kolíky spojené nie sú, táto funkcia vyrobí novú škatuľku (takú, ako by vyrobil magický tester) a jej návratovou hodnotou bude číslo tejto novej škatuľky. Ak teda dostaneš návratovú hodnotu väčšiu ako a, tester odpovedal "nie" a vyrobil novú škatuľku.

Akonáhle tvoj program dokončí volania funkcie `Connected`, musíš ešte graderu odovzdať riešenie. To spravíš zavolaním nasledujúcej funkcie gradera:

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

Jediným parametrom tejto funkcie je vektor `result`. Jeho prvky sú dvojice čísel. Každá takáto dvojica čísel popisuje jeden drôt. Takto odovzdaná sada drôtov musí popisovať škatuľku, ktorá je ekvivalentná so škatuľkou 0.

Vektor `result` nesmie obsahovať žiaden drôt, ktorý by mal oba konce na tom istom kolíku. Taktiež nesmie obsahovať žiadne dva drôty, ktoré by priamo prepájali tú istú (neusporiadanú) dvojicu kolíkov.

Grader spracuje takto odovzdaný výsledok a následne ukončí beh celého programu.

Constraints

- $2 \leq n \leq 200$

Scoring

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

Sample Interaction

Tvoj program zavolal	Grader zavolal alebo vrátil	Vysvetlenie
	`ToyDesign(4, 5)`	Škatuľka má 4 kolíky. Môžeš nanajvýš 5-krát zavolať funkciu `Connected`. Pomocou týchto volaní máš zistiť, ako vyzerá škatuľka 0.
`Connected(0, 1, 2)`	`return 1`	V škatuľke 0 platí, že kolíky 1

		a 2 nie sú spojené. Magický tester preto vyrobil novú škatuľku s číslom 1, v ktorej už spojené sú.
`Connected(1, 3, 2)`	`return 2`	V škatuľke 1 platí, že kolíky 3 a 2 nie sú spojené. Magický tester preto vyrobil zo škatuľky 1 novú škatuľku 2.
`Connected(0, 3, 4)`	`return 0`	V škatuľke 0 platí, že kolíky 3 a 4 sú vodivo spojené. Žiadna nová škatuľka teda v tomto kroku nevznikla.
`DescribeDesign({{3, 4}})`	–	Odovzdáme riešenie: škatuľku obsahujúcu jediný drôt, a to medzi kolíkmi 3 a 4.

Sample Grader

V súbore `ToyDesign.zip`, ktorý dostanete, nájdete ukážkový grader: `grader.cpp`

Tento grader začne tým, že načíta vstup zo štandardného vstupu, a to v nasledovnom formáte:

- V prvom riadku je počet kolíkov n a počet drôtov m.
- Nasleduje m riadkov a v každom z nich čísla dvoch kolíkov prepojených priamym drôtom.

Následne tento grader zavolá tvoju funkciu `ToyDesign`.

Skompilovať ho dokopy s tvojim riešením môžeš v termináli nasledovným príkazom:

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

Vo vyššie uvedenom príkaze `solution.cpp` je názov súboru s implementáciou tvojej funkcie (teda súbor, ktorý odovzdávaš do CMS).

Spustiť skompilovaný program na vstupe, ktorý je tiež v balíčku, vieš nasledovne:

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