Дизајн на играчка

Име на задачата	ToyDesign
Input File	Interactive Task
Output File	Interactive Task
Time limit	1 second
Memory limit	256 megabytes

Во компанија за дизајн на играчки е креирана нова играчка која личи вака: има n иглички, нумерирани со 1 до n, кои стрчат (излегуваат) од кутија. Некои од игличките се поврзани меѓусебно со жички, ама внатре во кутијата. (Со други зборови, игличките и жиците формираат ненасочен граф, каде игличките се темињата, а жиците се ребрата.) Жиците не се гледаат однадвор, а за да се дознае нешто за пар од иглички мора да се користи **тестер** на игличките: Може да избереме две иглички i и j такви што $i \neq j$, и тестерот ќе каже дали тие иглички се поврзани, директно или индиректно. (Што значи, тестерот кажува дали има пат во графот меѓу тие две темиња.)

Множеството врски во кутијата ќе го наречеме **дизајн** на играчката.

Користите специјализиран софтвер за прашување (query) и креирање на овие дизајни. Овој софтвер работи вака: Започнува со некој дизајн на играчката што го означуваме како „дизајн 0“. Тој, не ви ги прикажува врските во кутијата за овој дизајн. Наместо тоа, можете повеќе пати да ја извршувате следната операција:

- Избирате број на дизајн a и два броја на иглички i и j така што $i \neq j$.
- Софтверот ви кажува што би се случило ако го користиме тестерот на тие две иглички. Со други зборови, ви кажува дали игличките i и j се (директно или индиректно) поврзани во дизајнот a.
- Исто така, ако игличките не биле директно или индиректно поврзани во дизајнот a, тогаш се создава нов дизајн кој ги има сите врски од дизајнот a плус една дополнителна директна врска помеѓу i и j. На овој дизајн му е даден следниот достапен број за дизајн. (Значи, првиот дизајн создаден на овој начин ќе има број 1, потоа број 2 и така натаму.) Забележете дека ова не го менува дизајнот a, туку само создава нов дизајн кој има дополнителна врска.

Вашата цел е да научите што е можно повеќе за дизајнот 0 со користење на оваа операција.

Забележете дека не е секогаш можно да се одреди точното множество врски за дизајнот 0, бидејќи не постои начин да се разликуваат директните и индиректните врски. На пример, разгледајте ги следните два дизајни со $n = 3$:


Тестерот би го пријавил кој било пар иглички како поврзани за двата дизајни, така што овие два дизајни нема да можеме да ги разликуваме користејќи го софтверот опишан погоре.

Вашата цел е да одредите кој било дизајн што е еквивалентен на дизајнот 0. Два дизајни се **еквивалентни** ако тестерот го пријави истиот резултат во двата дизајни за сите пара иглички.

Implementation

Ова е интерактивен проблем. Мора да имплементирате функција

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

што одредува дизајн кој е *еквивалентен* на дизајнот 0. Вашата имплементација треба да ја постигне оваа цел со извршување на две конкретни операции нула или повеќе пати. Во една операција, вашата програма треба да ја повика функцијата:

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

каде што $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$ и a не смеат да го надминат бројот на досега креирани дизајни. Ако иглички i и j се (директно или индиректно) поврзани во дизајнот a, тогаш таа ќе врати a назад. Во спротивно, ќе го врати бројот на креирани дизајни до сега плус еден, што станува број доделен на новиот дизајн кој ги има сите врски на дизајнот a плус врската помеѓу i и j. Функцијата `Connected` може да се повика најмногу `max_ops` пати.

Кога вашата програма завршила со операциите `Connected`, таа треба да опише дизајн што е еквивалентен на дизајнот 0 и потоа да го заврши нејзиното извршување на стандарден начин. За да се опише дизајн, програмата треба да повика:

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

Параметарот `result` е вектор од парови од цели броеви кои ги опишуваат паровите иглички кои се директно поврзани. Секој пар одговара на една врска и треба да ги содржи двата броја на меѓусебно поврзани иглички. Мора да има најмногу една директна врска помеѓу секој (неподреден) пар иглички и да нема директни врски од игла до самата себе.

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

Contestant action	Grader action	Explanation
	`ToyDesign(4, 5)`	There are 4 pins in the toy. You need to determine any design that is equivalent to design 0 by calling `Connected` at most 5 times.
`Connect(0, 1, 2)`	`Returns 1.`	Pins 1 and 2 are not connected directly or indirectly in design 0. New design 1 is created.
`Connected(1, 3, 2)`	`Returns 2.`	Pins 3 and 2 are not connected directly or indirectly in design 1. New design 2 is created.
`Connected(0, 3, 4)`	`Returns 0.`	Pins 3 and 4 are connected directly or indirectly in design 0. No new design is created.
`DescribeDesign({{3, 4}})`	-	We describe a design that has only one connection: Pins 3 and 4.

Sample Grader

The provided sample grader, `grader.cpp`, in the task attachment `ToyDesign.zip`, reads the input from the standard input in the following format:

- The first line contains the number of nodes, n, the number of edges, m and *max_ops*
- The following m lines contain edges as node tuples.

The sample grader reads the input and calls the `ToyDesign` function in user's solution.

To compile the sample grader with your solution, you may use the following command at the terminal prompt:

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

where `solution.cpp` is your solution file to be submitted to CMS. To run the program with the sample input provided in the attachment, type the following command from at the terminal prompt:

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