Toy Design

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

Tu lucrezi pentru o companie care proiectează jucării. O jucărie nouă care este creată funcționează astfel: Există n ace, numerotate de la 1 la n, ieşind afară dintr-o cutie. Unele perechi de ace sunt conectate cu fire în interiorul cutiei. (Cu alte cuvinte, acele și firele formează un graf neorientat, unde acele sunt nodurile și firele sunt muchiile.) Firele nu sunt vizibile din exterior și singurul mod de a afla ceva legat de ele este de a folosi un **tester** asupra acelor: Putem alege două ace i și j astfel încât $i \neq j$, și tester-ul va spune dacă acele două ace sunt conectate în interiorul cutiei, fie direct, fie indirect. (Prin urmare, tester-ul spune dacă există un lanț între acele ace în graf.)

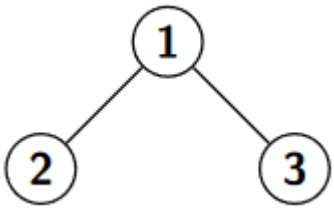
Vom numi setul de conexiuni din interiorul cutiei **design-ul** jucăriei.

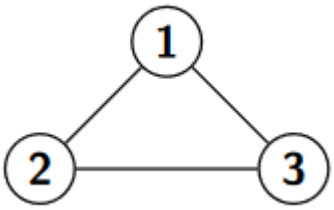
Tu folosești un software specializat pentru a interoga și crea aceste design-uri. Acest software funționează astfel: Începe cu un design al jucăriei la care ne referim ca "design 0". Nu îți va arăta conexiunile din interiorul cutiei pentru acest design. În schimb, tu poți aplica în mod repetat următoarea operație în trei pași:

1. Tu alegi un design numerotat a și două ace numerotate i și j astfel încât $i \neq j$.
2. Software-ul îți spune ce se va întâmpla dacă folosim tester-ul pe acele două ace. Cu alte cuvinte, îți spune dacă acele i și j sunt (direct sau indirect) conectate în design-ul a.
3. De asemenea, dacă acele nu au fost direct sau indirect conectate în design-ul a, îți creează un nou design care are toate conexiunile din design-ul a plus o conexiune directă suplimentară între i și j. Acestui design îi este dat următorul număr de design disponibil. (Deci, primul design creat în acest mod are numărul 1, apoi numărul 2, și așa mai departe.) Rețineți că aceasta nu schimbă design-ul a, doar creează un nou design care are conexiunea suplimentară.

Scopul tău este să afli cât mai mult posibil despre design-ul 0 folosind această operație.

Rețineți că nu este mereu posibil să se determine setul exact de conexiuni pentru design-ul 0, deoarece nu există un mod de a diferenția conexiunile directe de cele indirecte. De exemplu, luați în considerare următoarele două design-uri cu $n = 3$:





Tester-ul va raporta orice pereche de ace ca fiind conectate pentru ambele design-uri, deci noi nu vom putea să le diferențiem folosind software-ul descris mai sus.

Scopul tău este de a determina orice design care este echivalent cu design-ul 0. Două design-uri sunt **echivalente** dacă tester-ul reportează aceleași rezultate în ambele design-uri pentru toate perechile de ace.

Implementare

Aceasta este o problema interactivă. Tu trebuie să implementezi o funcție

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

care determină un design care este *echivalent* cu design-ul 0. Implementarea ta trebuie să atingă acest scop prin apelarea a două funcții după cum este descris mai jos. Prima funcție pe care o poți apela este:

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

unde $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, și a nu trebuie să depășească numărul de design-uri create până acum. Dacă acele i și j sunt (direct sau indirect) conectate în design-ul a, atunci va returna a. În caz contrar, va returna numărul de design-uri create până acum plus unu, care devine numărul atribuit noului design care are toate conexiunile design-ului a plus conexiunea dintre i și j. Funcția `Connected` poate fi apelată cel mult de `max_ops` ori.

Când programul tău a terminat cu operațiile `Connected`, ar trebui să descrie un design care este echivalent cu design-ul 0. Pentru a descrie un design, programul trebuie să apeleze:

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

Parametrul `result` este un vector de perechi de întregi ce descriu conexiunile directe dintre ace. Fiecare pereche corespunde cu o conexiune și trebuie să conțină cele două numere de ace din conexiune. Trebuie să fie cel mult o conexiune directă între fiecare pereche (neordonată) de ace, și

nicio conexiune directă între un ac și el însuși. Apelul acestei funcții termină execuția programului tău.

Restricții

- $2 \leq n \leq 200$

Punctaj

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

Exemplu de interacțiune

Acțiunea concurentului	Acțiunea grader-ului	Explicație
	`ToyDesign(4, 20)`	Sunt 4 ace în jucărie. Tu trebuie să determini orice design care este echivalent cu design-ul 0 apelând `Connected` cel mult de 20 ori.
`Connected(0, 1, 2)`	`Returnează 1.`	Acele 1 și 2 nu sunt conectate direct sau indirect în design-ul 0. Un nou design 1 este creat.
`Connected(1, 3, 2)`	`Returnează 2.`	Acele 3 și 2 nu sunt conectate direct sau indirect în design-ul 1. Un nou design 2 este creat.
`Connected(0, 3, 4)`	`Returnează 0.`	Acele 3 și 4 sunt conectate direct sau indirect în design-ul 0. Niciun nou design nu este creat.
`DescribeDesign({{3, 4}})`	-	Descriem un design ce are doar o conexiune: Acele 3 și 4.

Grader exemplu

Grader-ul exemplu furnizat, `grader.cpp`, în atașamentul problemei `ToyDesign.zip`, citește datele de intrare din intrarea standard în următorul format:

- Prima linie conține numărul de ace, n, numărul de conexiuni directe, m și *max_ops*
- Următoarele m linii conțin conexiunile directe ca perechi de ace.

Grader-ul exemplu citește intrarea și apelează funcția `ToyDesign` în soluția utilizatorului. Grader-ul va afișa unul dintre următoarele mesaje, depinzând pe comportarea soluției tale:

- "Wrong answer: Number of operations exceeds the limit.", dacă numărul de apeluri ale funcției `Connected` depășește *max_ops*
- "Wrong answer: Wrong design id.", dacă parametrul a din apelul lui `Connected` este numărul unui design care nu există în momentul în care este efectuat apelul.
- "Wrong answer: Incorrect design.", dacă design-ul descris prin `DescribeDesign` nu este echivalent cu design 0.
- "OK!" dacă design-ul descris prin `DescribeDesign` este echivalent cu design 0.

Pentru a compila grader-ul exemplu cu soluția ta, tu poți folosi următoarea comandă în promptul terminalului:

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

unde `solution.cpp` este fișierul soluției tale ce va fi trimisă pe CMS. Pentru a rula programul cu intrarea furnizată ca exemplu în atașament, tastează următoarea comandă în promptul terminalului:

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