Toy Design

Aufgabenname	ToyDesign
Eingabedatei	Interaktive Aufgabe
Ausgabedatei	Interaktive Aufgabe
Zeitlimit	1 Sekunde
Speicherlimit	256 Megabytes

Du arbeitest für ein Unternehmen, das Spielzeug entwickelt. Ein neues Spielzeug, das gerade entwickelt wird, funktioniert wie folgt: Es gibt n Pins, nummeriert von 1 bis n, die aus einer Schachtel herausragen. Einige Paare von Pins sind mit Drähten im Inneren der Schachtel verbunden. Mit anderen Worten, die Pins und die Drähte bilden einen ungerichteten Graphen, in dem die Pins die Knoten und die Drähte die Kanten sind. Die Drähte sind von aussen nicht sichtbar und die einzige Möglichkeit, etwas über sie herauszufinden, besteht darin, einen **Tester** an den Pins zu benutzen: Wir können zwei Pins i und j so auswählen (mit $i \neq j$) und der Tester wird feststellen, ob diese beiden Pins im Inneren der Box verbunden sind, entweder direkt oder indirekt. Der Tester sagt also, ob es einen Pfad zwischen diesen Pins im Graphen gibt.

Wir bezeichnen die Gesamtheit der Verbindungen in der Schachtel als das **Design** des Spielzeugs.

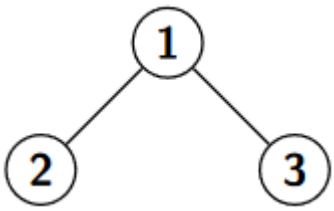
Du benutzt eine spezielle Software, um diese Muster abzufragen und zu erstellen. Diese Software funktioniert folgendermassen: Sie beginnt mit einem Design des Spielzeugs, das wir als "Design 0" bezeichnen. Sie zeigt dir die Verbindungen innerhalb der Schachtel für diesen Entwurf nicht an. Stattdessen kannst du wiederholt die folgende dreistufige Operation durchführen:

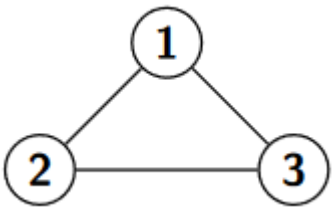
1. Du wählst eine Designnummer a und zwei Pin-Nummern i und j, sodass $i \neq j$.
2. Die Software sagt dir, was passieren würde, wenn du den Tester an diesen beiden Pins verwenden würdest. Mit anderen Worten, sie sagt dir, ob die Pins i und j im Entwurf a (direkt oder indirekt) verbunden sind.
3. Wenn die Pins im Entwurf a weder direkt noch indirekt verbunden waren, wird ein neuer Entwurf erstellt, der alle Verbindungen des Entwurfs a plus eine zusätzliche direkte Verbindung zwischen i und j enthält. Dieser Entwurf erhält die nächste verfügbare Designnummer. (Das erste auf diese Weise erstellte Design hat also die Nummer 1, dann die

Nummer 2 und so weiter). Beachte, dass dies das Design a nicht verändert, sondern nur ein neues Design mit der zusätzlichen Verbindung erzeugt.

Dein Ziel ist es, so viel wie möglich über das Muster 0 zu erfahren, indem du diese Operation anwendest.

Beachte, dass es nicht immer möglich ist, die genaue Menge der Verbindungen für Design 0 zu bestimmen, da es keine Möglichkeit gibt, direkte und indirekte Verbindungen zu unterscheiden. Zum Beispiel die folgenden zwei Designs mit $n = 3$:





Der Tester würde jedes Pin-Paar für beide Designs als verbunden melden, so dass wir nicht in der Lage sind, sie mit der oben beschriebenen Software zu unterscheiden.

Dein Ziel ist es, ein Design zu bestimmen, das äquivalent zu Design 0 ist. Zwei Designs sind **äquivalent**, wenn das Testgerät für alle Paare von Pins in beiden Designs das gleiche Ergebnis meldet.

Implementierung

Dies ist ein interaktives Problem. Du musst die Funktion

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

implementieren, die ein Design bestimmt, das *äquivalent* zu Design 0 ist. Deine Implementierung sollte dieses Ziel erreichen, indem sie die folgenden zwei Funktionen aufruft:

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

wobei $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$ und a die Anzahl der bisher erstellten Muster nicht überschreiten darf. Wenn die Pins i und j im Entwurf a (direkt oder indirekt) miteinander verbunden sind, dann wird a zurückgegeben. Andernfalls wird die Anzahl der bisher erstellten Designs plus eins zurückgegeben. Dies ist dann die Nummer, die dem neuen Design zugewiesen wird, das alle Verbindungen des Designs a plus die Verbindung zwischen i und j enthält. Die Funktion `Connected` kann maximal `max_ops` mal aufgerufen werden.

Wenn dein Programm mit den `Connected`-Operationen fertig ist, sollte es ein Design beschreiben, das dem Design 0 entspricht und dann seine Ausführung ordnungsgemäss beenden. Um ein

Design zu beschreiben, sollte das Programm folgende Funktion aufrufen:

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

Der Parameter `result` ist ein Vektor von Ganzzahlpaaren, welche die direkten Verbindungen zwischen den Pins beschreiben. Jedes Paar entspricht einer Verbindung und sollte die beiden Pin-Nummern der Verbindung enthalten. Es darf höchstens eine direkte Verbindung zwischen jedem (ungeordneten) Paar von Pins bestehen, und keine direkten Verbindungen zwischen einem Pin und sich selbst. Ein Aufruf dieser Funktion beendet die Ausführung deines Programmes.

Beschränkungen

- $2 \leq n \leq 200$.

Teilaufgaben

- Teilaufgabe 1 (10 Punkte): $n \leq 200$, *max_ops* $= 20'000$.
- Teilaufgabe 2 (20 Punkte): $n \leq 8$, *max_ops* $= 20$.
- Teilaufgabe 3 (35 Punkte): $n \leq 200$, *max_ops* $= 2'000$.
- Teilaufgabe 4 (35 Punkte): $n \leq 200$, *max_ops* $= 1'350$.

Beispiel-Interaktion

Aktion der Teilnehmerin	Aktion des Graders	Erklärung
	`ToyDesign(4, 20)`	Es sind 4 Pins in diesem Spielzeug. Du musst ein Muster ermitteln, das dem Muster 0 entspricht, indem du `Connected` höchstens 20 Mal aufrufst.
`Connected(0, 1, 2)`	`Returns 1.`	Die Pins 1 und 2 sind im Design 0 weder direkt noch indirekt verbunden. Es wird ein neues Design 1 erstellt.
`Connected(1, 3, 2)`	`Returns 2.`	Die Pins 3 und 2 sind in Design 1 weder direkt noch indirekt verbunden. Es wird ein neues Design 2 erstellt.
`Connected(0, 3, 4)`	`Returns 0.`	Die Pins 3 und 4 sind im Design 0 direkt oder indirekt verbunden. Es wird kein neues Design erstellt.

`DescribeDesign({{3, 4}})`	–	Wir beschreiben ein Design, das nur einen Anschluss zwischen den Pins 3 und 4 hat.

Beispielgrader

Der bereitgestellte Beispielgrader `grader.cpp` im Anhang `ToyDesign.zip` liest die Eingabe von der Standardeingabe in folgendem Format:

- Die erste Zeile enthält die Anzahl der Pins n die Anzahl der direkten Verbindungen m und *max_ops*.
- Die folgenden m Zeilen enthalten die direkten Verbindungen als Pin-Tupel.

Der Beispielgrader liest die Eingabe und ruft die `ToyDesign`-Funktion in deiner Lösung auf. Der Beispielgrader gibt abhängig vom Verhalten deiner Lösung eine der folgenden Fehlermeldungen aus:

- "Wrong answer: Number of operations exceeds the limit.", falls die Anzahl der Aufrufe der `Connected` Funktion *max_ops* überschreitet.
- "Wrong answer: Wrong design id.", falls der Parameter a im Aufruf der `Connected` Funktion im Moment des Aufrufes noch nicht existiert.
- "Wrong answer: Incorrect design.", falls das Design, das mit der Funktion `DescribeDesign` beschrieben wurde, nicht äquivalent zum Design 0 ist.
- "OK!", falls das Design, das mit der Funktion `DescribeDesign` beschrieben wurde, äquivalent zum Design 0 ist.

Um den Beispielgrader mit deiner Lösung zu kompilieren, kannst du den folgenden Befehl im Terminal verwenden:

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

wobei `solution.cpp` deine Lösungsdatei ist, die an das CMS übermittelt werden soll. Um das Programm mit der Beispieleingabe im Anhang auszuführen, gib den folgenden Befehl im Terminal ein:

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