Toy Design

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

Du arbetar för ett företag som säljer leksaker. Att skapa en ny leksak går till så här: Det finns n stycken pins, numrerade från 1 till n som sticker ut från en låda. Vissa par av pins är kopplade med ledningar inuti lådan. (Med andra ord, pins och ledningar bildar en oriktad graf där pins är noder och ledningar är kanter.) Ledningarna är inte synliga från utsidan och det enda sättet att ta reda på något om dem är genom att använda en **testare** på pins: Vi väljer två pins i och j så att $i \neq j$, och testaren kommer visa om dessa två pins är kopplade inuti lådan, antingen direkt eller indirekt. (Alltså berättar testaren om det finns en väg mellan dessa två pins i grafen.)

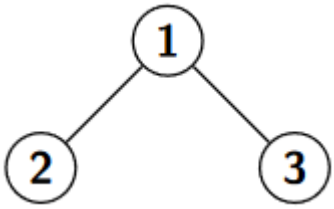
Vi kallar mängden kopplingar inuti lådan för **designen** av leksaken.

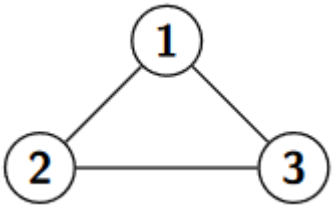
Du använder ett speciellt program för att inspektera och skapa dessa designer. Detta program fungerar så här: Det startar med en design av leksaken som vi kallar "design 0". Programmet visar inte kopplingarna inuti lådan för denna design. Istället kan du upprepade gånger utföra följande operation:

- Du väljer ett design-nummer a och två pin-nummer i och j så att $i \neq j$.
- Programmet berättar för dig vad som skulle hända om du använder testaren på dessa två pins. Med andra ord, det berättar för dig om pins i och j är (direkt eller indirekt) kopplade i design a.
- Också, om dessa pins inte var direkt eller indirekt kopplade i design a, kommer programmet skapa en ny design som har alla kopplingar från design a plus en extra direkt koppling mellan i och j. Denna design ges de nästa tillgängliga design-numret. (Så, den första designen som skapas på detta sätt kommer ha nummer 1, sedan nummer 2, och så vidare) Notera att detta inte ändrar design a, utan bara skapar en ny design som har en extra koppling.

Ditt mål är att lära dig så mycket som möjligt om design 0 genom att använda denna operation.

Notera att det inte alltid är möjligt att bestämma den exakta mängden kopplingar för design 0, eftersom det inte finns något sätt att särskilja direkta och indirekta kopplingar. Till exempel, betrakta följande två designer med $n = 3$:





Testaren skulle rapportera alla par av pins som kopplade i båda designerna, så vi skulle inte kunna särskilja dem med programmet som beskrevs ovan.

Ditt mål är att hitta någon design som är ekvivalent med design 0. Två designer är **ekvivalenta** om testaren rapporterar samma resultat i båda designer för alla par av pins.

Implementation

Detta är ett interaktivt problem. Du behöver implementera en funktion

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

som hittar en design som är *ekvivalent* med design 0. Din implementation ska uppnå detta genom att anropa funktioner som beskrivet nedan. Den första funktionen ditt program kan anropa är:

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

där $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, och a får inte vara större än antalet designer skapare än så länge. Om pins i och j är (direkt eller indirekt) kopplade i design a kommer funktionen returnera a. Annars kommer funktionen returnera antalet designer skapade än så länge plus ett, vilket blir numret associerat med den nya designen som har alla kopplingar från design a plus den extra kopplingen från i till j. Funktionen `Connected` kan anropas max `max_ops` gånger.

När ditt program är klart med `Connected`-operationerna, ska det beskriva en design som är ekvivalent med design 0. För att beskriva en design ska programmet anropa:

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

Parametern `result` är en vector av heltalspar som beskriver kopplingar mellan pins. Varje par beskriver en koppling och ska innehålla de två pins som kopplingen går mellan. Det måste finnas som mest en direkt koppling mellan varje oordnat par av pins, och ingen direkt koppling mellan en pin och sig själv. Att anropa denna funktion kommer avsluta exekveringen av ditt program.

Begräningar

- $2 \leq n \leq 200$

Poängsättning

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

Exempelinteraktion

Du gör något	Grader gör något	Förklaring
	`ToyDesign(4, 20)`	Det finns 4 pins i denna leksak. Du behöver hitta någon design som är ekvivalent till design 0 genom att anropa `Connected` som mest 20 gånger.
`Connected(0, 1, 2)`	`Returnerar 1.`	Pins 1 och 2 är inte kopplade direkt eller indirekt i design 0. Ny design 1 skapas.
`Connected(1, 3, 2)`	`Returnerar 2.`	Pins 3 och 2 är inte kopplade direkt eller indirekt i design 1. Ny design 2 skapas.
`Connected(0, 3, 4)`	`Returnerar 0.`	Pins 3 och 4 är kopplade direkt eller indirekt i design 0. Ingen ny design skapas.
`DescribeDesign({{3, 4}})`	-	Vi beskriver en design som bara har en koppling: Mellan pins 3 och 4.

Sample Grader

Den givna sample-gradern, `grader.cpp`, i bilagan `ToyDesign.zip`, läser indata från standard input i följande format:

- Den första raden innehåller antalet pins, n, antalet direkta kopplingar m och *max_ops*.
- De följande m raderna innehåller direkta kopplingar som tupler av pins.

Sample-gradern läser indatan och anropar `ToyDesign`-funktionen i den användarens lösning. Gradern skriver ut ett av följande meddelande, baserat på beteendet av din lösning:

- "Wrong answer: Number of operations exceeds the limit.", om antalet anrop till `Connected` övverstiger *max_ops*
- "Wrong answer: Wrong design id.", om parametern a till ett anrop av `Connected` är numret för en design som inte fanns när funktionen anropades.
- "Wrong answer: Incorrect design.", om designen som beskrevs med `DescribeDesign` inte är ekvivalent med design 0.
- "OK!" om designen som beskrevs med `DescribeDesign` är equivalent med design 0.

För att kompilera sample-gradern med din lösning kan du använda följande kommando i terminalen:

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

där `solution.cpp` är din lösningsfil som du kommer skicka in till CMS. För att köra programmet med exempelindatan som finns i bilagan, skriv följande kommando i terminalen:

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