Toy Design

Probleemnaam	ToyDesign
Invoerbestand	Interactive Task
Uitvoerbestand	Interactive Task
Tijdslimiet	1 second
Geheugenlimiet	256 megabytes

Je werkt voor een bedrijf dat speelgoed ontwerpt. Een nieuw stuk speelgoed dat jullie ontwerpen werkt als volgt. Uit een doos steken n pinnen, genummerd van 1 tot en met n. Sommige paren van pinnen zijn verbonden met touwtjes in de doos. (In andere woorden, de pinnen en de touwtjes vormen een undirected graph, met de pinnen als knopen en de touwtjes als lijnen.) De touwtjes zijn onzichtbaar van buiten, en je kunt alleen dingen te weten komen over de touwtjes door het gebruik van een **tester** op de pinnen: we kunnen 2 pinnen i en j oppakken als $i \neq j$ en de tester zegt of de pinnen in de doos direct of indirect verbonden zijn. (Dus, de tester zegt of er een pad is tussen deze pinnen in de graaf.)

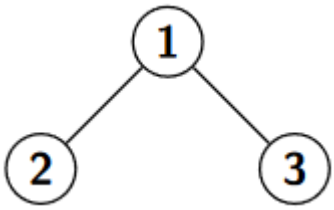
We noemen de set connecties in de doos het **design** van de doos.

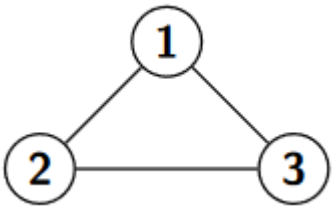
Je gebruikt gespecialiseerde software om deze designs te bevragen en te maken. De software werkt als volgt: het start met een design dat we "design 0" noemen. Het laat je de connecties in de doos niet zien. Maar je kunt wel herhaaldelijk de volgende 3 operaties uitvoeren:

1. Kies een design a en twee pinnen i en j met $i \neq j$.
2. De software zegt wat er gebeurt als we de tester gebruiken op deze twee pinnen. In andere woorden, het vertelt je of pin i en pin j (direct of indirect) verbonden zijn in design a.
3. Als de pinnen niet verbonden waren in design a, dan creeërt deze operatie ook een nieuw design, met alle connecties van design a plus één extra directe connectie tussen i en j. Dit design krijgt het eerstvolgende beschikbare nummer. (Het eerste design dat op deze manier gecreëerd wordt krijgt nummer 1, dan nummer 2, enzovoort.) Merk op dat dit niets verandert aan design a, het creeërt enkel een nieuw design met de extra connectie.

Jouw doel is om zoveel mogelijk te leren van design 0 door deze operatie te gebruiken.

Let op. Het is niet altijd mogelijk om de exacte set van connecties voor design 0 te bepalen, omdat er geen manier is om directe en indirecte connecties te onderscheiden. Bijvoorbeeld, bekijk de volgende twee designs met $n = 3$:





De tester zou alle paren van pinnen als verbonden teruggeven, dus we kunnen bovenstaande configuraties niet onderscheiden met de software zoals deze hierboven beschreven is.

Jouw doel is om een design te bepalen, dat equivalent is aan design 0. Twee designs zijn **equivalent** als de tester hetzelfde resultaat geeft in beide designs voor alle paren pinnen.

Implementatie

Dit is een interactief probleem. Implementeer de functie

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

die een design bepaalt dat *equivalent* is aan design 0. Je implementatie moet dit bereiken door twee specifieke functies aan te roepen zoals hieronder beschreven. De eerste functie die jouw programma kan aanroepen is:

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

waar $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, en a niet groter is dan het aantal gecreëerde designs tot nu toe. Als pinnen i en j verbonden zijn in design a (direct of indirect), dan krijg je a terug. Anders krijg je het aantal tot nu toe gecreëerde designs plus één terug. Dit is het nummer van het nieuw-gecreëerde design, met alle connecties uit design a plus de connectie tussen i en j. De functie `Connected` mag maximaal `max_ops` keer aangeroepen worden.

Als jouw functie klaar is met de `Connected` operaties, dan moet het een design opleveren dat equivalent is aan design 0. Roep

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

aan om het design te beschrijven. De parameter `result` is een vector of integer pairs die de directe connecties tussen pinnen representeert. Elk paar beschrijft één connectie door de twee pin-indexen te noemen van de connectie. Er mag maximaal één directe connectie bestaan tussen

elk paar pinnen, en geen directe connecties tussen een pin en zichzelf. Deze functie aanroepen beëindigt jouw programma.

Randvoorwaarden

- $2 \leq n \leq 200$

Puntentelling

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

Voorbeeld interactie

Deelnemer actie	Grader actie	Toelichting
	`ToyDesign(4, 5)`	Er zitten 4 pinnen in de doos. Je moet een design equivalent aan design 0 bepalen door maximaal 5 keer `Connected` aan te roepen.
`Connect(0, 1, 2)`	`Returns 1.`	Pinnen 1 en 2 zijn niet direct of indirect verbonden in design 0. Design 1 is gecreëerd.
`Connected(1, 3, 2)`	`Returns 2.`	Pinnen 3 en 2 zijn niet direct of indirect verbonden in design 1. Design 2 is gecreëerd.
`Connected(0, 3, 4)`	`Returns 0.`	Pinnen 3 en 4 zijn direct of indirect verbonden in design 0. Er wordt geen nieuw design gecreëerd.
`DescribeDesign({{3, 4}})`	-	We beschrijven een design met één connectie: Pinnen 3 en 4.

Voorbeeld Grader

De gegeven voorbeeld grader, `grader.cpp`, in de bijlage van de taak `ToyDesign.zip`, leest de input van de standaard input in het volgende formaat:

- Op de eerste regel staan het aantal pinnen n , het aantal directe verbindingen m en het maximum aantal operations *max_ops*.
- De volgende m lijnen bevatten de directe verbindingen als tuples van pinnen.

De voorbeeld grader leest de input en roept de functie `ToyDesign` aan in jouw oplossing. De grader geeft als output een van de volgende berichten, gebaseerd op het gedrag van jouw oplossing:

- "Wrong answer: Number of operations exceeds the limit.", wannneer het aantal aanroepen naar `Connected` *max_ops* overschrijdt.
- "Wrong answer: Wrong design id.", wanneer de parameter a in een aanroep naar `Connected` een cijfer is van een design wat niet bestond op het moment dat de aanroep is uitgevoerd.
- "Wrong answer: Incorrect design.", wanneer het design omschreven door `DescribeDesign` niet equivalent is aan design 0.
- "OK!" wanneer het design beschreven door `DescribeDesign` equivalent is aan design 0.

Gebruik het volgende commando aan in de terminal prompt, om de voorbeeld grader met jouw oplossing te compileren:

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

waar `solution.cpp` jouw oplossing is zoals je die gaat oploaden in het CMS. Gebruik het volgende commando in de terminal prompt, om het programma te runnen met de voorbeeld input in de bijlage:

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