Toy Design

Feladat neve	ToyDesign
Input File	Interaktív feladat
Output File	Interaktív feladat
Időkorlát	1 másodperc
Memóriakorlát	256 MB

Egy játékokat tervező vállalatnál dolgozol, ahol egy új játékot készítetek: Egy dobozból n tű áll ki, amelyeket 1-től n-ig megszámoztak. Néhány tűpár a doboz belsejében dróttal össze van kötve. (Más szóval a tűk és a drótok egy irányítatlan gráfot alkotnak, ahol a tűk a csúcsok, a drótok pedig az élek.) A drótok kívülről nem láthatók, és az egyetlen módja annak, hogy megtudjunk róluk valamit, ha **teszteljük** őket: kiválasztunk két tűt, i-t és j-t úgy, hogy $i \neq j$, és a teszter megadja, hogy ez a két tű közvetlenül vagy közvetve összekapcsolódik-e a dobozon belül. (Tehát a teszter azt mondja meg, hogy van-e út a gráfban ezen tűk között.)

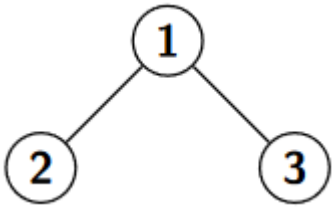
A dobozon belül a kapcsolatok összességét a játék **mintá**jának nevezzük.

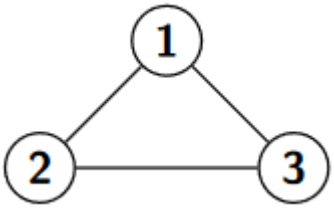
Ezeknek a mintáknak a lekérdezéséhez és megtervezéséhez egy speciális szoftvert használsz . Ez a szoftver úgy működik, hogy a játék valamelyik mintájával indul, amelyet "0. mintának" nevezünk. A szoftver nem mutatja meg neked a minta dobozon belüli kapcsolatait, de a következő három lépésből álló műveletet végezheted el vele:

1. Kiválasztasz egy a mintaszámot és két tűt i-t és j-t úgy, hogy $i \neq j$.
2. A szoftver megmondja, hogy mi történne, ha a tesztert ezen a két tűn használnánk. Vagyis megmondja, hogy az i és j tűk (közvetlenül vagy közvetve) összekapcsolódnak-e az a mintában.
3. Ha a tűk nem voltak közvetlenül vagy közvetve összekötve az a mintában, akkor létrehoz egy új mintát, amely az összes kapcsolatot tartalmazza az a mintából, plusz egy további közvetlen kapcsolatot az i és j között. Ez a minta kapja a következő elérhető mintaszámot. (Tehát az első ilyen módon létrehozott minta az 1-es, a következő a 2-es számot kapja, és így tovább.) Mindez nem változtatja meg az a mintát, csak egy új mintát hoz létre, amelyben van egy új drót (közvetlen kapcsolat).

A művelet segítségével kell minél többet megtudnod a 0. mintáról.

Megjegyezzük, hogy nem mindig lehetséges a 0. minta kapcsolatainak pontos meghatározása, mivel nem lehet megkülönböztetni a közvetlen és közvetett kapcsolatokat. Vegyük például a következő két tervet $n = 3$ értékkel:





A teszter mindkét kialakításnál bármelyik pár tűt összekapcsoltnak jelzi, így a fent leírt szoftverrel nem tudjuk megkülönböztetni a két elrendezést.

A feladatod meghatározni egy mintát, ami a 0. mintával egyenértékű. Két minta akkor **egyenértékű**, ha a teszter mindkét mintánál minden tűpárra ugyanazt az eredményt adja .

Megvalósítás

Ez egy interaktív feladat. A

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

eljárást kell megvalósítanod, amely egy olyan mintát határoz meg, amely *egyenértékű* a 0. mintával. A megvalósításhoz az alább leírt két függvényt kell alkalmaznod. Az első függvény, amit használhatsz, az az

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

ahol $1 \leq i, j \leq n, i \neq j, a \geq 0$ és a nem haladhatja meg az eddig létrehozott minták számát. Ha az i és j tűk (közvetlenül vagy közvetve) összekapcsolódnak az a mintában, akkor a-t fog visszaadni. Egyébként pedig az eddig létrehozott minták száma plusz egyet ad vissza, ami az új minta száma lesz, és amelyben az a terv összes csatlakozása megtalálható, valamint az i és j tűk közötti közvetlen kapcsolatot is tartalmazza. A `Connected` függvényt legfeljebb `max_ops` alkalommal lehet meghívni.

Amikor a programod befejezi a `Connected` műveleteket, egy olyan mintát kell megadnia, amely egyenértékű a 0. mintával. A minta leírásához a programnak meg kell hívnia az alábbi eljárást:

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

A `result` paraméter egy egész számpárokból álló vektor, amely a tűk közötti közvetlen kapcsolatokat adja meg. Minden számpár egy kapcsolatnak felel meg és a két összekapcsolt tű

sorszámát tartalmazza. Minden (rendezetlen) tűpár között legfeljebb egy közvetlen kapcsolat lehet, és nem lehet közvetlen kapcsolata egy tűnek önmagával. Ennek az eljárásnak a meghívása után a program futtatása befejeződik.

Korlátok

- $2 \leq n \leq 200$

Pontozás

1. részfeladat (10 pont): $n \leq 200$, *max_ops* $= 20\,000$
2. részfeladat (20 pont): $n \leq 8$, *max_ops* $= 20$
3. részfeladat (35 pont): $n \leq 200$, *max_ops* $= 2\,000$
4. részfeladat (35 pont): $n \leq 200$, *max_ops* $= 1\,350$

Minta interakció

A programod lépése	Az értékelő lépése	Magyarázat
	`ToyDesign(4, 20)`	4 tű van a játékban. A `Connected` legfeljebb 5 alkalommal történő meghívásával kell meghatározni minden olyan mintát, amely egyenértékű a 0. mintával.
`Connected(0, 1, 2)`	`Returns 1.`	A 0. mintában az 1. és 2. tűk nincsenek összekapcsolva sem közvetlenül, sem közvetve. Új, 1. minta jön létre.
`Connected(1, 3, 2)`	`Returns 2.`	Az 1. mintában a 3. és 2. tűk nincsenek összekapcsolva sem közvetlenül, sem közvetve. Új, 2. minta jön létre.
`Connected(0, 3, 4)`	`Returns 0.`	A 0. mintában a 3. és 4. tű össze van kapcsolva közvetlenül vagy közvetve. Nem jön létre új minta.
`DescribeDesign({{3, 4}})`	-	Egy olyan mintát írunk le, amely csak egy kapcsolattal

		rendelkezik: a 3. és 4. tű között.

Sample Grader

A `grader.cpp` mintaértékelő a `ToyDesign.zip` csatolmányban található. A standard bemenetről olvas be a következő formátumban:

- Az első sor tartalmazza a tűk n számát, a drótok m számát és a *max_ops* értéket.
- A következő m sor tartalmazza a tesztelt tűk sorszámait..

A mintaértékelő beolvassa a bemenetet és meghívja a `ToyDesign` függvényt.

Az értékelő a megoldástól függően a következő üzenetek egyikét fogja kiírni:

- "Wrong answer: Number of operations exceeds the limit.", ha a `Connected` függvény hívása meghaladja a *max_ops* értéket.
- "Wrong answer: Wrong design id.", ha a meghívott a paraméter egy olyan minta száma, ami a hívás pillanatában nem létezik.
- "Wrong answer: Incorrect design.", ha a `DescribeDesign` eljárással leírt minta nem egyenértékű a 0. mintával.
- "OK!", ha a `DescribeDesign` eljárással leírt minta megegyezik a 0. mintával.

A mintaértékelő megoldásoddal együtt való futtatásához parancssorból használhatod a

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

parancsot, ahol a `solution.cpp` a te megoldásod, amit majd a CMS-be töltesz fel. A mellékletben szereplő mintabemenettel futtathatod a programod, ha a parancssorba a

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

parancsot írod.