おもちゃの設計 (Toy Design)

問題名	おもちゃの設計
入力ファイル	(インタラクティブ課題)
出力ファイル	(インタラクティブ課題)
実行時間制限	1秒
メモリ制限	256MB

あなたはおもちゃを考案する企業で働いている．考案中の新しいおもちゃは次のようになっている: 1 から n までの番号の付けられた n 個のピンが箱からでている．いくつかのピンの組は箱の内部で針金でつながっている．(すなわち，ピンと針金は，ピンを頂点，針金を辺とする無向グラフになっている．)

針金は外側からは見えず，針金について何かを知る唯一の方法は，**テスター** をピンに使用することである: $i \neq j$を満たす 2 つのピン i と j を選んでテスターを使用すると，その 2 つのピンが箱の内部で，直接的または間接的につながっているかを知ることができる．(つまり，テスターはグラフ上でその 2 つのピンの間にパスが存在するかを教えてくれる．)

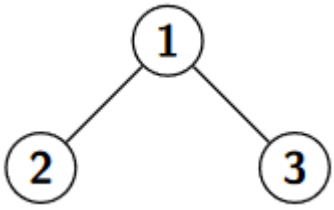
これから，箱の内部でのピンのつながりを，おもちゃの **設計** と呼ぶ．

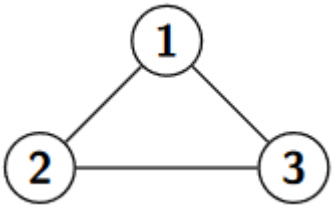
あなたはこれらの設計を参照しながら作成するために，特殊なソフトウェアを使っている．このソフトウェアは次のように動く: まず，あるおもちゃの設計から始めてこれを "設計 0" とする．この設計のピンのつながりは表示されない．代わりにあなたは次の 3 段階の操作を繰り返し行うことができる:

1. 設計番号 a と $i \neq j$ を満たす 2 つのピン番号 i，j を選ぶ．
2. ソフトウェアはその 2 つのピンにテスターを使用した結果を報告する．言い換えると，設計 a においてピン i とピン j が (直接的または間接的に) つながっているかを知ることができる．
3. もし設計 a において選んだ 2 つのピンが直接的または間接的につながっていなかった場合，ソフトウェアは設計 a に含まれるすべての針金に，ピン i とピン j をつなぐ針金を加えた新たな設計を作る．この設計には次に使用可能な設計番号が与えられる．(つまり，このように作られた最初の設計は番号 1 を持ち，次は番号 2 という風に続いていく．) このとき，針金が加わった新たな設計が作られるだけで，設計 a は変わらないことに注意せよ．

あなたの目標はこの操作を使用して，設計 0 の内部をできるだけ特定することである．

直接的なつながりと間接的なつながりを判別できないため，設計 0 の内部のつながりを必ずしも特定できるとは限らないことに注意せよ．例えば，$n = 3$ の次の 2 つの設計を考える．





両方の設計とも，すべてのピンの組について，テスターはつながっていると報告する．よって上述のソフトウェアを用いてこれらを判別することはできない．

あなたの目標は設計 0 と等価な設計を決定することである．2 つの設計は，すべてのピンの組について，テスターが同じ結果を報告するときに限り **等価** であるとする．

実装の詳細

これはインタラクティブ課題である．あなたは関数

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

を実装しなければならない．この関数は設計 0 と *等価* な設計を決定しなければならない．あなたは後述の 2 つの関数を呼び出すことでこれを行う．呼び出すことのできる 1 つ目の関数を次に示す．

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

$1 \leq i, j \leq n, i \neq j, a \geq 0$ を満たし，かつ a はこれまでに作られた設計の数を超えてはならない．もし設計 a においてピン i と ピン j が (直接的または間接的に) つながっていた場合，関数は a を返す．そうでなければ，これまでに作られた設計の個数に 1 を加えた数を返す．この数は，設計 a に含まれるすべての針金に，ピン i とピン j をつなぐ針金を加えた新たな設計の番号となる．関数 `Connected` は高々 `max_ops` 回まで呼び出せる．

あなたのプログラムが `Connected` 操作を終えたなら，設計 0 と等価な設計を示さなければならない．設計を示すためには，次の関数を呼び出す．

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

引数 `result` はピンの間を直接つなぐ針金を表す，整数のペア型の配列である．各ペアは 1 つの針金に対応しており，ピンの番号を表す 2 つの数でなければならない．各ピンの組について，それらをつなぐ針金は高々 1 つでなければならない．また，あるピンとそれ自身が直接つながるような針金があってはならない．この関数の呼び出しであなたのプログラムの実行は終了する．

制約

- $2 \leq n \leq 200$

小課題

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

例

あなた	ジャッジシステム	説明
	`ToyDesign(4, 20)`	おもちゃには 4 つのピンがある．あなたは，たかだか 20 回の `Connected` の呼び出しで，設計 0 と等価な設計を決定する必要がある．
`Connected(0, 1, 2)`	`Returns 1.`	設計 0 において，ピン 1 とピン 2 は直接的または間接的につながっていない．新たに設計 1 が作られる．
`Connected(1, 3, 2)`	`Returns 2.`	設計 1 において，ピン 3 とピン 2 は直接的または間接的につながっていない．新たに設計 2 が作られる．
`Connected(0, 3, 4)`	`Returns 0.`	設計 0 においてピン 3 とピン 4 は直接的または間接的につながっている．新たに設計は作られない．
`DescribeDesign({{3, 4}})`	-	ピン 3 とピン 4 のつながり 1 つだけを持つ設計を示す．

採点プログラムのサンプル (Sample Grader)

配布された `ToyDesign.zip` の中に入っている，採点プログラムのサンプル `grader.cpp` は，入力を以下の形式で標準入力から受け取る．

- 1行目にはピンの数 n と針金の数 m そして *max_ops* が書かれている．
- 続く m 行には辺の情報が 2 つの頂点の組として書かれている．

採点プログラムのサンプルは，入力を受け取った後あなたの解答プログラムの `ToyDesign` 関数を呼び出す．採点プログラムのサンプルは，あなたの解答プログラムの挙動に応じて，以下のメッセージの内いずれか 1 つを出力する．

- "Wrong answer: Number of operations exceeds the limit."，関数 `Connected` の呼び出し回数が *max_ops* を超えたとき．
- "Wrong answer: Wrong design id."，関数 `Connected` を呼び出す際，引数 a が関数を呼び出した時点では存在しない設計の番号であったとき．
- "Wrong answer: Incorrect design."，関数 `DescribeDesign` で示された設計が設計 0 と等価でなかったとき．
- "OK!"，関数 `DescribeDesign` で示された設計が設計 0 と等価であったとき．

採点プログラムのサンプルと，あなたの解答プログラムをコンパイルする方法の一つとして，Terminal 上で以下のコマンドを打つという手がある．

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

ここで `solution.cpp` は，CMS上で提出されるべき解答プログラムである．また，配布されたファイルの中に入っている入出力例を用いてプログラムを実行する方法の一つとして，Terminal 上で以下のコマンドを打つという手がある．

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