航空路線図 (Airline Route Map)

JOI 国に住む Alice は，IOI 国に住む Bob を招待することにし，それに先立って JOI 国内の航空路線図を送ることにした．JOI 国は，島 0 から島 $N-1$ までの N 個の島からなる島国である．JOI 国には M 本の航空路線があり，$i+1$ 番目の航空路線は $(0 \leqq i \leqq M-1)$，島 A_i と島 B_i を双方向に結んでいる．同じ 2 つの島を結ぶ，異なる 2 つの航空路線は存在しない．JOI 国から IOI 国に情報を送るには，JOI 国の運営する特殊な電報を使う．この電報では無向グラフを送ることができる．ただし，頂点番号と辺番号はランダムにシャッフルされてしまう．

具体的には，次のようにして情報が送られる．Alice が送るグラフを G とする．(G の頂点数を V，辺数を U とする．)

- Alice が G の頂点数 V と辺数 U を指定する．また，G の各頂点に $0, 1, \ldots, V-1$ の番号を，G の各辺に $0, 1, \ldots, U-1$ の番号をそれぞれ付ける．
- Alice はパラメータ $C_0, C_1, \ldots, C_{U-1}$ および $D_0, D_1, \ldots, D_{U-1}$ を指定する．このパラメータは G の辺を表す．つまり，各 j $(0 \leqq j \leqq U-1)$ について，G の辺 j は 頂点 C_j と頂点 D_j を結ぶ．
- JOI 国によって，G の頂点番号がシャッフルされる．まず，JOI 国は $0, 1, \ldots, V-1$ の並べ替えである数列 $p[0], p[1], \ldots, p[V-1]$ を生成する．次に，$C_0, C_1, \ldots, C_{U-1}$ を $p[C_0], p[C_1], \ldots, p[C_{U-1}]$ に書き換え，$D_0, D_1, \ldots, D_{U-1}$ を $p[D_0], p[D_1], \ldots, p[D_{U-1}]$ に書き換える．
- 続いて，JOI 国によって，G の辺番号がシャッフルされる．まず，JOI 国は $0, 1, \ldots, U-1$ の並べ替えである数列 $q[0], q[1], \ldots, q[U-1]$ を生成する．次に，$C_0, C_1, \ldots, C_{U-1}$ を $C_{q[0]}, C_{q[1]}, \ldots, C_{q[U-1]}$ に書き換え，$D_0, D_1, \ldots, D_{U-1}$ を $D_{q[0]}, D_{q[1]}, \ldots, D_{q[U-1]}$ に書き換える．
- V と U の値と，書き換えられた後のパラメータ $C_0, C_1, \ldots, C_{U-1}$ および $D_0, D_1, \ldots, D_{U-1}$ が Bob に伝えられる．

ただし，この電報で送ることのできるグラフは，単純グラフのみである．単純グラフとは，多重辺や自己ループを含まないグラフのことである．つまり，各 i, j $(0 \leqq i < j \leqq U-1)$ について $(C_i, D_i) \neq (C_j, D_j)$ かつ $(C_i, D_i) \neq (D_j, C_j)$ を満たし，さらに全ての i $(0 \leqq i \leqq U-1)$ について $C_i \neq D_i$ を満たすグラフのみ送ることができる．

Alice は，できるだけ少ない頂点数のグラフで Bob に JOI 国内の航空路線図を送りたい．

課題

Alice と Bob の通信を実現するために，次の 2 つのプログラムを作成せよ．

- 1 つ目のプログラムは，JOI 国の島の数 N，JOI 国の航空路線の本数 M，JOI 国の航空路線を表す数列 A, B が与えられたとき，Alice が送信するグラフ G の情報を設定する．
- 2 つ目のプログラムは，Bob が受信するグラフ G の情報が与えられたとき，JOI 国の航空路線図を復元する．

実装の詳細

あなたは 2 つのファイルを提出しなければならない．

1 つ目のファイルは `Alice.cpp` という名前である．このファイルは Alice が送信するグラフを設定するファイルであり，以下のルーチンを実装していなければならない．プログラムは `Alicelib.h` をインクルードすること．

- `void Alice( int N, int M, int A[], int B[] )`

  この関数は各テストケースにおいて 1 回だけ呼び出される．

  - 引数 `N` は JOI 国の島の数を表す．
  - 引数 `M` は JOI 国の航空路線の本数を表す．
  - 引数 `A[]`, `B[]` は長さ M の数列であり，JOI 国の航空路線を表す．

  関数 `Alice` 中では以下の関数を呼び出すことで，送信するグラフ G の情報を設定する．

  ★ `void InitG( int V, int U )`

  この関数は，G の頂点数と辺数を設定する．

  - ◇ 引数 `V` は G の頂点数を表す．`V` は 1 以上 1500 以下の整数値でなければならない．この範囲外の値を指定して呼び出した場合，**不正解 [1]** と判定される．
  - ◇ 引数 `U` は G の辺数を表す．`U` は 0 以上 $V(V-1)/2$ 以下の整数値でなければならない．この範囲外の値を指定して呼び出した場合，**不正解 [2]** と判定される．

  ★ `void MakeG( int pos, int C, int D )`

  この関数は，G の辺を設定する．

  - ◇ 引数 `pos` は 設定する辺の番号を表す．`pos` は 0 以上 $U-1$ 以下の整数値でなければならない．この範囲外の値を指定して呼び出した場合，**不正解 [3]** と判定される．また, 同じ引数 `pos` で 2 回以上呼び出すことはできない．同じ引数で 2 回呼び出した場合，**不正解 [4]** と判定される．
  - ◇ 引数 `C` および `D` は G の辺 `pos` の両端の頂点を表す．`C` および `D` は 0 以上 $V-1$ 以下の整数値でなければならない．また，$C \neq D$ を満たさなければならない．`C` または `D` がこれらの条件を満たさないとき，**不正解 [5]** と判定される．

  ただし，U および V は `InitG` で設定した値である．

  関数 `Alice` 中では，関数 `InitG` を 1 回呼び出した後，関数 `MakeG` をちょうど U 回呼び出さなければならない．関数 `InitG` が 2 回以上呼び出された場合, **不正解 [6]** と判定される．関数 `InitG` が呼び出される前に, 関数 `MakeG` が呼び出された場合, **不正解 [7]** と判定される．関数 `Alice` の実行の終了時に関数 `InitG` が呼び出されていない場合や，関数 `MakeG` の呼び出し回数が U でなかった場合, **不正解 [8]** と判定される．関数 `Alice` の実行の終了時に指定されたグラフ G が単純グラフでなかった場合，**不正解 [9]** と判定される．

  `Alice` の呼び出しが不正解と判定された場合，その時点でプログラムは終了する．

2 つ目のファイルは Bob.cpp という名前である．このファイルは，Bob が受け取ったグラフ G から JOI 国の航空路線図を表すグラフを復元する方法を実装したファイルであり，以下のルーチンを実装していなければならない．プログラムは Boblib.h をインクルードすること．

- `void Bob( int V, int U, int C[], int D[] )`

  この関数は各テストケースにおいて 1 回だけ呼び出される．

  - 引数 V はグラフ G の頂点数を表す．
  - 引数 U はグラフ G の辺数を表す．
  - 引数 C[], D[] は長さ U の数列であり，グラフ G の辺を表す．

  関数 Bob 中では以下の関数を呼び出すことで，復元した JOI 国の航空路線図の情報を記述する．

  ★ `void InitMap( int N, int M )`

  この関数を呼び出すことで，JOI 国の島の数と JOI 国の航空路線の本数を記述する．

  - ◇ 引数 N は復元した JOI 国の島の数を表す．N は整数値であり，実際の JOI 国の島の数と一致していなければならない．一致していない場合，不正解 **[10]** と判定される．
  - ◇ 引数 M は復元した JOI 国の航空路線の本数を表す．M は整数値であり，実際の JOI 国の航空路線の本数と一致していなければならない．一致していない場合，不正解 **[11]** と判定される．

  ★ `void MakeMap( int A, int B )`

  この関数を呼び出すことで，JOI 国の航空路線を記述する．

  - ◇ 引数 A および B は JOI 国に島 A と島 B を結ぶ航空路線があることを表す．A および B は 0 以上 $N-1$ 以下の整数値でなければならない．また，$A \neq B$ を満たさなければならない．A または B がこれらの条件を満たさないとき，不正解 **[12]** と判定される．JOI 国に島 A と島 B を結ぶ航空路線がないとき，不正解 **[13]** と判定される．また，過去の呼び出しと同じ航空路線を表す呼び出しをしてはならない．MakeMap(A, B) の呼び出し時に，既に MakeMap(A, B) または MakeMap(B, A) を実行していたとき，不正解 **[14]** と判定される．

  ただし，N は InitMap で設定した値である．

  関数 Bob 中では，関数 InitMap を 1 回呼び出した後，関数 MakeMap をちょうど M 回呼び出さなければならない．関数 InitMap が 2 回以上呼び出された場合，不正解 **[15]** と判定される．関数 InitMap が呼び出される前に，関数 MakeMap が呼び出された場合，不正解 **[16]** と判定される．関数 Bob の実行の終了時に関数 InitMap が呼び出されていない場合や，関数 MakeMap の呼び出し回数が M でなかった場合，不正解 **[17]** と判定される．ただし，M は InitMap で設定した値である．

  Bob の呼び出しが不正解と判定された場合，その時点でプログラムは終了する．

採点の手順

採点は以下の手順で行われる．不正解と判定された場合はその時点でプログラムは終了される．

(1) JOI 国の航空路線図を表す情報を引数として，関数 `Alice` を 1 回呼び出す．

(2) 関数 `Alice` 中で設定されたグラフを G とする．G の頂点番号と辺番号をシャッフルしたグラフを表す情報を引数として，関数 `Bob` を 1 回呼び出す．

(3) 採点される．

重要な注意

- 内部での使用のために他のルーチンを実装したり，グローバル変数を宣言するのは自由である．ただし，提出された 2 つのプログラムは, 採点プログラムとまとめてリンクされて 1 つの実行ファイルになるので，各ファイル内のすべてのグローバル変数と内部ルーチンを `static` で宣言して，他のファイルとの干渉を避ける必要がある．採点時には，このプログラムは Alice 側, Bob 側として 2 個のプロセスとして実行されるので，Alice 側と Bob 側でプログラム中のグローバル変数を共有することはできない．
- あなたの提出したプログラムは，標準入力・標準出力，あるいは他のファイルといかなる方法でもやりとりしてはならない．ただし，標準エラー出力にデバッグ情報等を出力することは許される．

コンパイル・実行の方法

作成したプログラムをテストするための，採点プログラムのサンプルが，コンテストサイトからダウンロードできるアーカイブの中に含まれている．このアーカイブには，提出しなければならないファイルのサンプルも含まれている．

採点プログラムのサンプルは 1 つのファイルからなる. そのファイルは `grader.cpp` である．作成したプログラムを `Alice.cpp` および `Bob.cpp` とする．作成したプログラムをテストするには，これらのファイル (`grader.cpp`, `Alice.cpp`, `Bob.cpp`) と，`Alicelib.h` および `Boblib.h` を同じディレクトリに置き，次のようにコマンドを実行する．

```
g++ -std=c++14 -O2 -o grader grader.cpp Alice.cpp Bob.cpp
```

コンパイルが成功すれば，`grader` という実行ファイルが生成される．

実際の採点プログラムは，採点プログラムのサンプルとは異なることに注意すること．採点プログラムのサンプルは単一のプロセスとして起動する．このプログラムは，標準入力から入力を読み込み，標準出力に結果を出力する．

採点プログラムのサンプルの入力

採点プログラムのサンプルは標準入力から以下の入力を読み込む．

- 1 行目には 2 つの整数 N, M が書かれている．N は JOI 国の島の数を，M は JOI 国の航空路線の本数を表す．
- 続く M 行には，M 本の航空路線の情報が書かれている．M 行のうちの $i+1$ 行目 $(0 \leqq i \leqq M-1)$ には，2 つの整数 A_i, B_i が書かれている．これは，JOI 国の航空路線の情報を表す．

採点プログラムのサンプルの出力

採点プログラムのサンプルは標準出力へ以下の情報を出力する（引用符は実際には出力されない）．

- プログラムの実行中にいずれかの不正解と判定された場合，不正解の種類が “`Wrong Answer [1]`” のように出力され，実行が終了される．
- `Alice` と `Bob` のいずれの呼び出しも不正解と判定されなかった場合，“`Accepted`” と出力される．また，V の値が出力される．

実行するプログラムが複数の不正解の条件を満たした場合，表示される不正解の種類はそれらのうち 1 つのみである．

制限

すべての入力データは以下の条件を満たす．

- $1 \leqq N \leqq 1\,000$.
- $1 \leqq M \leqq N(N-1)/2$.
- $0 \leqq A_i \leqq N-1$ $(0 \leqq i \leqq M-1)$.
- $0 \leqq B_i \leqq N-1$ $(0 \leqq i \leqq M-1)$.
- $A_i \neq B_i$ $(0 \leqq i \leqq M-1)$.
- $(A_i, B_i) \neq (A_j, B_j)$ かつ $(A_i, B_i) \neq (B_j, A_j)$ $(0 \leqq i < j \leqq M-1)$.

小課題

この課題では小課題は全部で 3 個ある．各小課題の配点および追加の制限は以下の通りである．

小課題 1 [22 点]

- $N \leqq 10$.

小課題 2 [15 点]

- $N \leqq 40$.

小課題 3 [63 点]

追加の制限はない．

採点基準

- 小課題 1 および小課題 2 では，各小課題のすべてのテストケースに正解すると，満点が与えられる．
- 小課題 3 では，すべてのテストケースに正解したとき，$V - N$ の値の最大値を MaxDiff として，以下のように採点される．
  - $101 \leqq$ MaxDiff のとき，0 点．
  - $21 \leqq$ MaxDiff $\leqq 100$ のとき，$13 + \left[\dfrac{100 - \text{MaxDiff}}{4}\right]$ 点．ここで，x を超えない最大の整数を $[x]$ で表す．
  - $13 \leqq$ MaxDiff $\leqq 20$ のとき，$33 + (20 - \text{MaxDiff}) \times 3$ 点．
  - MaxDiff $\leqq 12$ のとき，63 点．

やり取りの例

grader が読み込む入力の例と，それに対応する関数の呼び出しの例を以下に示す．

入力例 1	ルーチンの呼び出しの例			
	呼び出し	戻り値	呼び出し	戻り値
4 3	`Alice(...)`			
0 1			`InitG(4,3)`	
0 2				(なし)
0 3			`MakeG(0,1)`	
				(なし)
			`MakeG(0,2)`	
				(なし)
			`MakeG(0,3)`	
				(なし)
		(なし)		
	`Bob(...)`			
			`InitMap(4,3)`	
				(なし)
			`MakeMap(0,1)`	
				(なし)
			`MakeMap(0,2)`	
				(なし)
			`MakeMap(0,3)`	
				(なし)
		(なし)		

このとき，`Alice(...)`，`Bob(...)` に渡される引数はそれぞれ，例えば次のようになる．

引数	`Alice(...)`	`Bob(...)`
`N`	4	
`M`	3	
`V`		4
`U`		3
`A`	$\{0, 0, 0\}$	
`B`	$\{1, 2, 3\}$	
`C`		$\{2, 2, 2\}$
`D`		$\{3, 0, 1\}$

入力例 2
5 7 0 1 0 2 1 3 1 4 3 4 2 3 2 4